VISUALIZATION BY DEMONSTRATION

A Dissertation
Presented to
The Academic Faculty

By

Bahador Saket

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology

August 2020

VISUALIZATION BY DEMONSTRATION

Approved by:

Dr. Alex Endert, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. John Stasko
School of Interactive Computing
Georgia Institute of Technology

Dr. Gregory Abowd
School of Interactive Computing
Georgia Institute of Technology

Dr. Keith Edwards
School of Interactive Computing
Georgia Institute of Technology

Dr. Niklas Elmqvist
College of Information Studies
University of Maryland

Date Approved: April 28, 2020

True knowledge exists in knowing that you know nothing.

Socrates

To my parents.

ACKNOWLEDGEMENTS

I would like to acknowledge the amazing people that have helped me along the way, both personally and professionally. I am fortunate to have such amazing people around me who have been instrumental in helping me to reach this point.

First, I want to thank me. I want to thank me for believing in me. I want to thank me for doing all this hard work. I want to thank me for having no days off over the past five years. I want to thank me for never quitting.

I would like to thank my parents for their love and support through the entire process. Thank you for encouraging me to follow my passion and instilling the confidence and optimism in me that has been really valuable through my graduate career. I understand being a parent is difficult, even more difficult is being a friend to your child. I am lucky that you were both my parent and a friend.

I would also like to thank my lovely wife, Farnaz. It is really hard to explain the influence you have had on me. Your kindness, confidence, and selfless desire to help others will always be an inspiration to me. Thank you for teaching me to dream BIG.

I next want to thank my advisor, Alex Endert. I want to acknowledge and appreciate all the work you do that has enabled me to reach this point of my graduate career. Thanks for being a mentor, colleague, and a great friend over the past five years.

I also wish to thank my thesis committee, John Stasko, Gregory Abowd, Keith Edwards, and Niklas Elmqvist, for your great advice on shaping this work. Each of you has been a great role model and I undoubtedly intend to use what I have learned from each of you as I move forward in my career.

I also want to thank James Foley, Rahul Basole, Munmun De Choudhury, Rosa Arriaga, Thomas Ploetz, Remco Chang, Stephen Kobourov, Carlos Scheidegger, Charles Perin, and Samuel Huron for providing feedback on my research and career plans.

I have also had the fortunate opportunity to have worked with great people outside of Georgia Tech through internships as a student. Thanks to Cagatay Demiralp, Shengdong Zhao, Koji Yatani, and Darren Edge for allowing me to work in your teams, which gave me a fresh new look at research.

I have been fortunate to be part of the Information Visualization lab. I would like to thank the current and past members. During my Ph.D., I have also been lucky to collaborate with many great people in UbiComp and SocWeB labs at Georgia Tech. You all have been great colleagues and friends. Working with you gave me fresh perspectives to research in Human-Computer Interaction.

Last but not least, I want to thank my best friend, Farshid Tavakolizadeh. You are a wonderful friend, and I appreciate your kindness, support, and generosity. Having a friend like you is one of the biggest chances of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

A key component of visualization systems that helps human sensemaking is interactivity. Thoughtfully designed interactions make the visual analysis process a conversation between the user and the interface that results in a deeper understanding of data. Yet, despite decades of research, existing visualization systems still require users to interact through layers of menus on control panels. These systems incur extra execution and cognitive cost by introducing a large number of intermediary interface elements such as menus and dialog boxes.

This dissertation proposes a novel interaction paradigm for visual data exploration called "visualization by demonstration". This paradigm aims to reduce the cognitive cost and enhance interaction expressivity to decrease the level of formalism and fundamental knowledge often required for visual data exploration. This dissertation first discusses the fundamental principles and guidelines that go into the design of visualization by demonstration. It then discusses how we can apply these fundamental principles and guidelines to design and develop general-purpose data visualization tools that implement visualization by demonstration. It finally applies visualization by demonstration to design and develop visual data exploration tools for experts in specific domains such as biology and healthcare.

CHAPTER 1
INTRODUCTION

Visual displays provide the highest bandwidth channel from computers to humans. In fact, a human absorbs more information through vision than through all other senses combined [1]. We often visualize abstract data to extend our ability to memorize, process, manipulate, and understand data. One of the main benefits of data visualization is the sheer quantity of information that can be rapidly interpreted if presented well [2].

Interaction is an essential part of data visualizations [3]. Without interaction, a data visualization becomes a static image. Interactivity can be added to data visualizations to engage users in visualization construction and data analysis processes. For example, interactive visualization tools might provide a set of features to enable users to interactively construct different visualizations (e.g., bar charts, scatterplots) based on the given data. Similarly, they can enable users to map data attributes to a variety of visual encodings (e.g., size or color of data points). The ultimate goals of many interactive visualization tools are either 1) to enable users to construct visualizations to convey their messages to an audience (e.g., communication and storytelling [4]) or 2) to gain insight and explore their data through an iterative process (e.g., exploratory data analysis [5]). The focus of this thesis is on visualization tools designed for exploratory data analysis.

A commonly-used interaction paradigm in visualization tools is *Manual View Specification (MVS)*. As the name implies, this interaction paradigm requires users to manually specify the desired visualization specifications and parameters through GUI operations typically designed as control panels. For instance, consider the process of interacting with data in a scatterplot. Users must specify data attributes to map onto axes, create any additional data mappings to encodings such as color, size, or shape, and finally set conditional filters to show only the relevant information. For situations where users' mental models contain

this level of detail and specificity, their tasks are well-supported by manual view specification. However, there exists tasks that are often harder to perform using MVS since 1) these tasks are ill-defined (it is nontrivial for users to break them down into a set of lower level operations on the control panel), or 2) they are repetitive or highly customized (require users to go through layers of menus).

For instance, consider the task of clustering data points showing patients diagnosed with HIV, where a biologist wants to cluster patients with HIV diseases based on their demographic factors (e.g., age and gender). In this case, the biologist can simply break the task down into two operations on the control panels (she first selects the clustering operation and then specifies the features that the data points should be cluster based on). However, there exists more ill-defined version of the clustering task. Consider a biologist who wants to cluster the patients. Based on her domain knowledge, she recognizes that some patients have to be in the same cluster because their share common symptoms and risk factors. In this case, it is really hard for the biologist to specify the exact parameters by which to cluster her data.

Such ill-defined tasks leave users in an ironic situation. The irony of the situation is that the burden of transforming the user's higher-level goals to the system's lower-level specifications and creating visualizations is carried by the user, not the system. This burden of specification hinders the data analysis process. It is a truism that computers are good at performing tedious tasks. So why is it that users need to go through a cumbersome process of specifying a variety of parameters through layers of menus?

In this dissertation, I investigated a novel interaction paradigm for visual data exploration called *Visualization by Demonstration (VbD)*. VbD allows users to provide partial demonstrations to the visual representation to indicate their intended changes. Using these demonstrations, the system first interprets users' intended changes and then applies/recommends potential change(s). Demonstrational interfaces trade off requiring users to specify visualization and system parameters for demonstrating the intended changes.

Taking the same clustering example from above, to cluster the patients that share the common symptoms and risk factors using VbD, the biologist could demonstrate how part of the expected clustering output should look like (e.g., she could drag patients that should belong to the same cluster close to each other), from which the system computes and recommends potential clustering results.

The trade-off between VbD and traditional MVS interfaces has multiple facets that designers must consider when determining if by-demonstration is suitable for the given task, visualization, or user group. For instance, VbD interfaces decrease intermediary graphical elements between users and systems. They reduce the need for users to translate their high level and sometimes ill-defined goals/tasks into a series of lower-level operations typically specified via control panels. Despite these advantages, challenges for demonstrational interfaces include how to correctly infer user's intentions from the given demonstrations, and how to make the potential space of operations discoverable to users. For example, coloring a data point in a scatterplot green could imply multiple meanings including coloring all data points green, mapping a data attribute to color encoding, color that specific data point green, and others.

I have to emphasize that not every task can be performed using the "by demonstration" approach. The MVS paradigm is still the easiest way to specify many well-specified and focused tasks, and will be so for many tasks supported in future systems. However, there exists tasks that can be expressed and performed using VbD, and the goal here is to identify classes of these tasks and investigate visualization by demonstration's effectiveness for expressing them.

1.1 Thesis Statement and Research Questions

Visualization by Demonstration (VbD) is an interaction paradigm for visual data exploration. This paradigm decreases the level of formalism and increases the level expressivity and flexibility in user interaction with data visualizations. VbD empowers the user during

the data analysis process by balancing the responsibility between the user and the system — users provide visual demonstrations, and based on this, the system infers potential specifications and then provides concrete recommendations.

- **RQ1: What are the fundamentals of the visualization by demonstration paradigm?** In Chapter 2, I define VbD and discuss different components that go into the systems implementing this paradigm. I also discuss challenges associated with each of the components in the VbD paradigm. Finally, I explain where VbD is effective or preferred over the existing MVS paradigm.

- **RQ2: How do people demonstrate their intended goals using direct manipulation of graphical encodings?** In Chapter 4, I first explain a study that investigates what type of direct manipulation strategies participants employ to visually demonstrate their intended changes. Based on findings of this study, I identify a list of visual demonstrations people employ to perform each operation and derive implications to help designers leverage direct manipulation of visual mark for providing demonstrations.

- **RQ3: How effective is manipulating graphical encodings used in visualizations as a method for providing visual demonstrations?** In Chapter 4, I then explain a study that investigates how fast and accurate participants can manipulate 12 different interactive visual marks (e.g., length). Results of this study showed that participants had above 80% accuracy in adjusting a majority of visual marks.

- **RQ4: How can we apply visualization by demonstration to design tools for data exploration?** In Chapter 5, I first explain how I designed and developed VisExemplar to show feasibility of the VbD paradigm. I used VisExemplar as testbed to identify operations that can be performed by demonstration.

- **RQ5: How effective is visualization by demonstration compare to traditional manual view specification?** In Chapter 5, I then explain a study that I conducted to empirically evaluate VisExemplar. I conducted a series of comparative studies using two tools: one implementing the commonly used MVS (Polestar) and another implementing VbD (VisExemplar). My goal was to investigate the trade-offs between these two paradigms.

- **RQ6: How to offer the benefits of both visualization by demonstration and manual view specification in a unified visualization tool?** In Chapter 5, I finally explain my process in designing and developing a tool called Liger. I designed Liger to investigate opportunities and challenges in combining two MVS and VbD paradigms in a unified tool. Through the design and implementation of Liger, I investigated how MVS and VbD paradigms can be blended to generate context that complements the individual paradigms.

- **RQ7: How can visualization by demonstration enable domain experts to perform real-world tasks such as data clustering?** In Chapter 6, I explain how I applied the VbD paradigm to design and develop a novel visual analysis tool designed to support cluster analysis for biologists. Geono-Cluster is currently deployed and used by biologists at Georgia Tech.

CHAPTER 2
VISUALIZATION BY DEMONSTRATION

RQ1: *What are the fundamentals of the visualization by demonstration paradigm?*

Conveying a process or an outcome to someone by demonstration might be one of the oldest forms of communicating one's knowledge and intentions in apprenticeship-style learning contexts. People are effective in communicating their intended goals and results by gesturing, drawing visuals, and other forms of demonstration to guide someone else through the intended results. In this thesis, I designed an interaction paradigm that extends the demonstrational paradigm to visual data exploration in a method called Visualization by Demonstration (VbD) [6]. VbD enables users to provide partial visual demonstrations to the visual representation to indicate their intention for more general changes. From these given demonstrations, the system interprets users' intentions, and recommends potential mappings and specifications. VbD aims to empower the user during the data analysis process by balancing the responsibility between the user and the system — users provide visual demonstrations, and based on this, the system infers potential specifications and then provides concrete recommendations.

Imagine an HIV researcher who wants to cluster her patients' dataset to derive meaningful patterns and relationships. By skimming through her data, she notices that two of her patients have the same cell decline level (CD4) of 67 (see Figure 7.1-Left). She wonders which other patients fall into the same group as these two patients. She drags the two data points representing these two patients onto the canvas to demonstrate her interest in finding cluster configurations that locate them together. Based on this demonstration, the system provides recommendations for different clustering results for the entire dataset (see Figure 7.1-Middle). Among all recommendations, she selects the recommendation that clusters the data based on combination of both gender and ethnicity features. The resulting

visualization converges with her prior knowledge that Hispanic males have an average CD4 level of 60 (see Figure 7.1-Right).

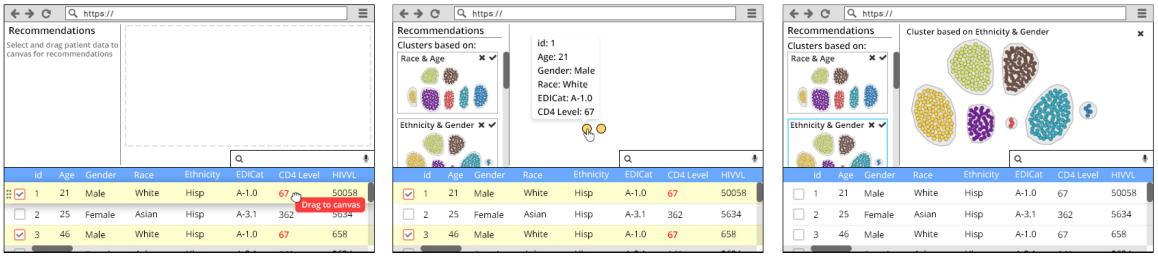
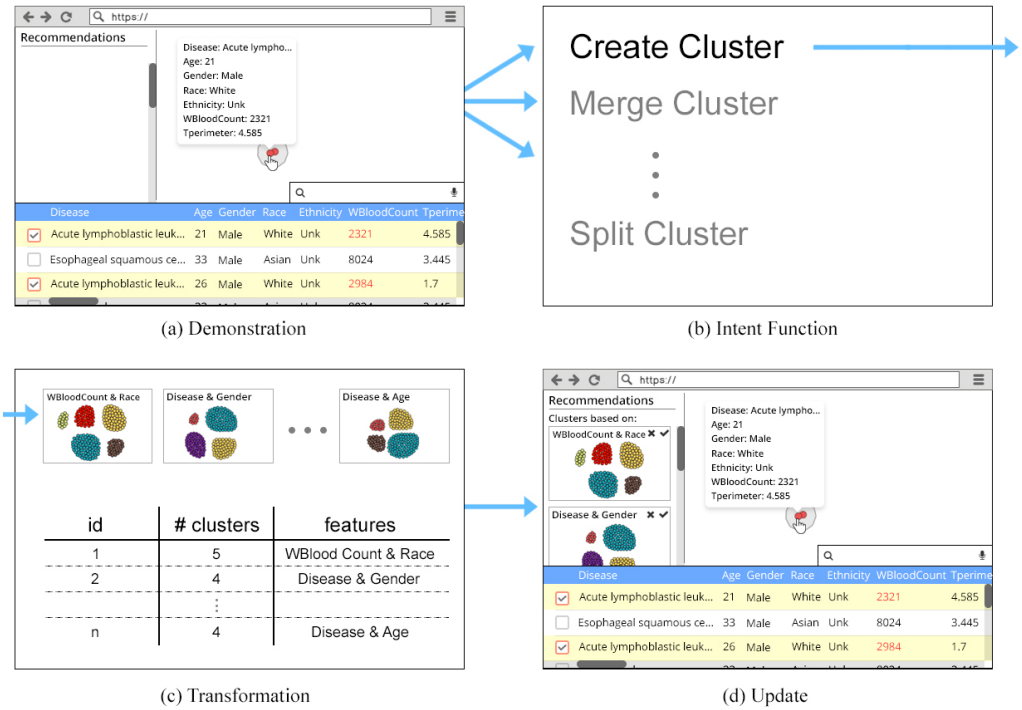


Figure 2.1: Left: The user selects a sample of data points and drags them to the canvas. **Middle:** Interface infers clustering results based on the given demonstration. **Right:** User selects a recommendation and gains insight.

Although this work is about VbD systems, for context it is useful to characterize a few non-demonstrational visualizations systems. Non-demonstrational visualization systems are missing the inferring component which is key to the visualization by demonstration systems. In other words, non-demonstrational interfaces do not infer the users' intentions from their interactions with visual representations. For example, visualization tools such as Tableau are not VbD systems because they execute the commands that the users are performing on the control panel in a sequential order. Tableau is not capable of interpreting/inferring the user's higher level expected results/changes based on her interactions. On the other hand, tools such as SketchStory [7] falls under the category of VbD tools mainly because it interprets the user's interest in assigning an attribute to an axis based on a simple axis that is sketched by a user.

2.1 Components of Visualization by Demonstration

VbD systems generally adhere to the process shown in Figure 2.2. This process includes four main components: **visual demonstrations**, **intent functions**, **transformation functions**, and **view update**. See Figure 2.2 for more details.



(a) Demonstration (b) Intent Function

(c) Transformation (d) Update

Figure 2.2: Conceptual diagram of demonstrational visual interfaces. **(a)** A user selects a subset of data points from the data table and drags them onto the canvas to demonstrate her interest in finding cluster configurations that locate them together. **(b)** Intent functions extracted multiple intentions including creating cluster, merging clusters, splitting clusters, and others. **(c)** Transformation functions computed top K possible clustering results and ranked them based on their relevance. **(d)** Possible transformations are recommended on the Interface.

2.1.1 Demonstrations

Each demonstration is a set of actions that a user takes to show parts of the expected results/changes visually (see Figure 2.2-a). VbD systems enable users to provide visual demonstrations to convey their partial intended results. For example, users might provide visual demonstrations by moving, coloring, or resizing a data point or changing the length of a bar in a bar chart, and others. Alternatively, as shown in Figure 2.2-a, the user selects a subset of data points from the data table and drags them onto the canvas to demonstrate her interest in finding cluster configurations that locate them together. Users might use one or more input modalities (e.g., sketch and touch) to provide visual demonstrations in systems implementing this paradigm. Figure 2.3 shows how users might provide different

demonstrations as a method for conveying their intended changes. Different input modalities could be used for providing visual demonstrations (e.g., touch, stylus, and mouse). However, this work mainly concentrates on investigating VbD systems on desktop devices using mouse input.

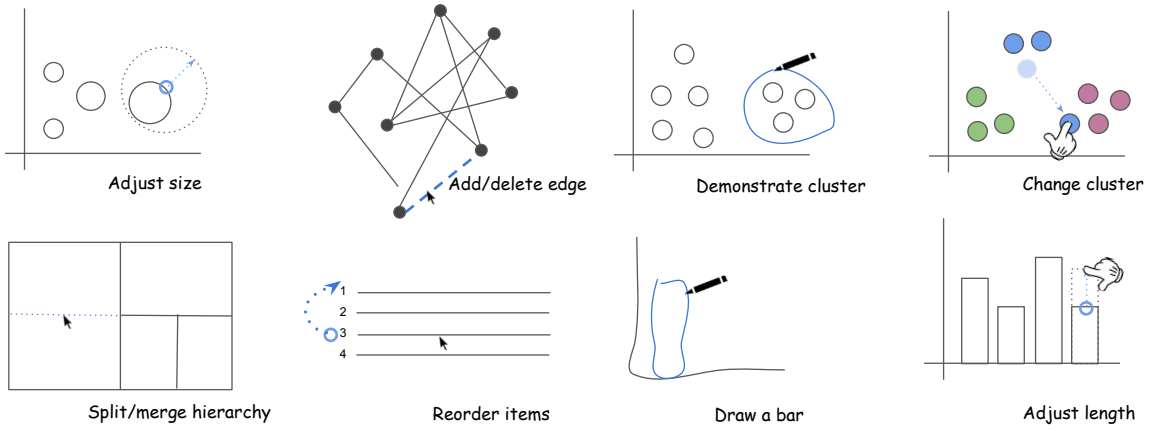


Figure 2.3: Visualization by demonstration can apply to many different visual representations and analytical tasks through different methods of providing visual demonstrations.

Different VbD systems have different starting points. Some VbD tools start with an empty canvas. For example, in SketchStory [7], users start by sketching on the empty canvas to demonstrate their intentions. In Geono-Cluster [8], users can drag a subset of data points from the data table and locate them close to each other on the empty canvas to demonstrate their interest in creating a cluter. Other VbD tools such as InterAxis [9], Podium [10], and VisExemplar [6] start with a visualization as a starting point. They enable users to provide visual demonstrations by manipulating the visual marks used in the visual representation. For example, VisExemplar [6] shows a set of circles that are randomly positioned on a 2D view (each circle represents a data point in the dataset) [6]. VisExemplar enables users to provide demonstrations by directly manipulating the visual marks used in the visual representation (e.g., directly manipulating the position, color, and size of circles).

Demonstrations are performed within visual metaphors. The ability for demonstrations to serve as visual analogies for the intended task will influence how effective they are. For

example, previous work showed that dragging the tallest bar in a bar chart to the extreme left or right of an axis is an intuitive way for users to demonstrate their interest in sorting the bar chart [11, 12]. However, there exist tasks that would be more difficult to demonstrate, or have more ambiguity in the system's interpretation. For instance, we recently found that the users had difficulties in finding an appropriate visual analogy to demonstrate their interest in switching from a scatterplot visualization to a bar chart using the mouse input [13]. Without intuitive and easy visual analogies to demonstrate an intended task or goal, the effectiveness of by-demonstration may suffer, and other interaction paradigms may be better suited (e.g., MVS). Alternatively, there may be demonstrations that are too ambiguous (i.e., the system could interpret the demonstration to mean too many different tasks). For instance, if a user moves only one data point in a scatterplot, what task does that demonstrate? It may reflect a desire to shift all points, change the scale on the axis, cluster similar points, change the axis, etc. In these situations, more demonstrations can decrease the ambiguity incrementally and help the system better interpret the given demonstration.

2.1.2 Intent Functions

When a user provides a demonstration, the demonstration interface calls a set of intent functions (see Figure 2.2-b). Intent functions are a set of rules that extract the user's intended goal/task based on the given demonstrations. Input to an intent function is details of the changes that are made to the visualization by users to demonstrate the intended goals. These details might include information about the visual glyphs that users interacted to provide visual demonstration. These information might include position, size, color, underlying bound data, and others. For example, if the user colors two data points in a scatterplot, the input to intent functions would be the id of the colored data points, their initial color, their new color, and data attributes bound to those data points. The output of an intent function is a list of potential user's intentions/goals that can be mapped to those demonstrations. For example, by coloring two data points blue in a scatterplot visualiza-

tion, intent functions would identify several potential intentions including coloring all data points blue, assigning a data attribute to the color of data points, and others.

2.1.3 Transformation Functions

Transformation functions are used to *compute* and *rank* the potential transformations that can be applied to the visualization given the set of demonstrations (see Figure 2.2-c). A transformation is moving from a visualization state to a different state by applying a set of visual changes to the visualization in the initial state.

Computation: First, a list of transformations is computed where, if applied, the final visual representation would match the given demonstration. For example, when demonstrating that two or more points should be in the same cluster, only clustering results that meet this constraint are shown.

Ranking: After computing a list of transformations, the system ranks these transformations based on their likelihood and fit. There are different ways to calculate the likelihood of a transformations. For example, VisExemplar updates the list of transformations after each interaction. Each transformation consists of a name and a likelihood. The likelihood value for each transformation indicates the number of times the transformation is generated. The likelihood value is normalized to a range of [0,1] and the table is updated to contain the normalized likelihood value for each transformation. The system only shows transformations with likelihood above 0.3. Visualization by demonstration systems might use different methods to calculate the likelihood of the potential transformations. After calculating the likelihood, they might suggest all or a subset of transformations.

The main difference between intent and transformation functions is that intent functions observe *"interaction types"* to extract the general intentions of the user. However, the transformation functions observe the *"interacted visual elements and data bounded to them"* to figure out how the state of the visualization should change. For example, coloring

two data points is enough of an evidence for the intent function to extract that the user might be interested in assigning a data attribute to color. But which data attribute should be assigned to color? As such, it is the transformation function's responsibility to figure out what data attributes are more appropriate to be mapped to color. In this case, transformation function searches to find what data attributes are common between the two data points that are colored blue. See Figure 2.2-(c) for more details.

2.1.4 Interface Updates

Once the potential meanings and possible transformations are extracted, the system decides how to apply/recommend possible transformations in the interface (see Figure 2.2-d). If there is one to one mapping between the given demonstrations and their potential meanings, the system might apply the changes directly in the interface (similar to InterAxis [9], Embedded Merge & Split [14], and Podium [10]). If there exist multiple meanings for the given demonstration, the system might apply the most possible intention and update the view accordingly, or recommend a set of possible meanings to the users in the interface (similar to VisExemplar [6] and Geono-Cluster [8]).

2.2 When is Visualization by Demonstration Beneficial?

A practical question that is immediately relevant when considering visualization by demonstration is – *when is it beneficial, effective, or preferred over existing interaction paradigms?* From a review of tasks supported from systems in relevant literature, as well as our own experiences designing by-demonstration systems, we discuss the tradeoffs associated with the demonstrational interaction paradigm along the following three factors.

2.2.1 Task Knowledge

Data visualization tasks range from low-level to high-level tasks, as discussed by Amar et al. [15]. In general, low-level tasks require fewer parameter specifications to perform compared to high-level tasks. However, in determining whether VbD or MVS is more effective, it is important to consider how many of these parameters that must be specified are not known to the user. In other words, if the task is well-defined (where all the parameters and their values are known) or ill-defined (where some of the parameters or their values are unknown) impacts the design decision about whether to support it with MVS or VbD. See Figure 2.4 for more details.

For instance, consider the task of filtering data points out of a scatterplot showing homes for sale, where a user wants to filter out homes with less than 3 bedrooms. In this case, there are 2 parameters that must be specified: the operation (filter) and the criteria (value of less than 3 for the variable 'bedrooms'). If both of these are known, MVS is an effective interaction paradigm to use. Users are capable of breaking such tasks down into a handful of operations on the control panels. However, even for relatively straightforward tasks such as filtering, more complex task alternatives may create situations where users do not know all the needed parameters. Taking the same example from above, what if the user instead wants to filter out homes similar to two or three she found and was not interested in? Further, she does not have enough clarity at the time to define what her interests are, and thus cannot specify the exact parameters by which to filter. Instead, she could demonstrate her intent to filter out specific points (e.g., she could demonstrate to the system that she is not interested in those homes by coloring or deleting them), from which the system computes and recommends potential filtering functions and parameter.

In another example, consider the task of clustering data points showing patients diagnosed with HIV, where a biologist wants to cluster patients with HIV diseases based on their demographic factors (e.g., Age and gender). In this case, the biologist can simply break the task down into two operations on the control panels (she first selects the clus-

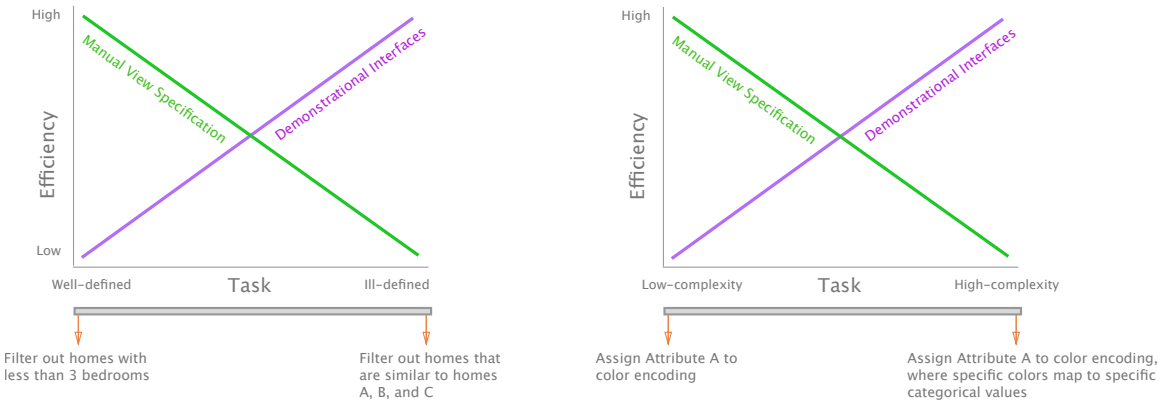


Figure 2.4: Task factors (e.g., task knowledge shown in the Left or task complexity shown on the right) influence the potential task effectiveness, and may help designers decide which interaction paradigm to use to support it.

tering operation and then specifies the features that the data points should be cluster based on). However, there exists more ill-defined version of the clustering task. Consider a biologist who wants to cluster the patients. Based on her domain knowledge, she recognizes that some patients have to be in the same cluster because their share common symptoms. In this case, it is really hard for the biologist to specify the exact parameters by which to cluster. Instead, she could demonstrate how part of the expected clustering output should look like (e.g., she could drag patients that should belong to the same cluster close to each other), from which the system computes and recommends potential clustering results.

In the context of user interface design, Myers [16] also discusses the difficulties of "by-demonstration" interfaces for cases where tasks are very specific and well-defined. Myers mentions *"demonstrational interfaces are harder to use in cases where the user knows exactly the relationship desired and could select it from a menu."* Additionally, he discusses that demonstrating well-defined tasks may be more time-consuming than selecting among a pre-defined set of controls in a menu [16].

2.2.2 Task Complexity

Tasks vary in complexity, based on factors including how many lower-level operations they can be broken down into [17]. This factor into the design decision about which interaction paradigm best supports these tasks (see Figure 2.4). The MVS paradigm can incur extra

execution and cognitive costs especially as the number of lower-level operations that a task can be broken into increases.

For example, consider commonly-used tasks, such as adjusting data grouping criteria (e.g., merging two bins in a histogram visualization). Currently, to perform this task in tools such as Tableau, users need to 1) select the variable and then select the Edit command from the pop-up menu. 2) In the Edit Bins dialogue, users can input new size for bins. 3) Users might also move to the next dialogue for further customization of binning. Sarvghad et al. [14] showed how users can demonstrate their interest in adjusting data grouping criteria by directly manipulating the visual glyphs. For instance, a user can drag and extend the width of a bar in a histogram to increase the range of the values presented by the bar. In response to the user's alteration of visualization, the system reconfigures new grouping of data values and reconstructs the view to new specifications. Sarvghad et al. [14] showed that the by demonstration paradigm can significantly reduce interaction time compared to the MVS alternatives.

In another example, imagine users want to map the data attribute A to the size encoding in a scatterplot, where the radius of the smallest circle is three. Currently, to perform this task in tools such as Tableau, users need to 1) map the attribute A to the size encoding. 2) Right-click on the size legend and select the "Edit Sizes" option. 3) A new window pops up that contains a variety of options including a slider to specify the minimum and maximum sizes. A demonstrational visualization interface could enable the users to perform the same task by changing the size of one or more data points to the expected value. As a result of this demonstration, the system could then recommend the transformation of adjusting the size of data points accordingly.

CHAPTER 3
RELATED WORK

In this chapter, I first discuss the definition of interaction in data visualization. I then explain different categories of desktop-based interactive visualization tools. I also explain MVS, the interaction paradigm that is implemented in many visualization tools today. I finally describe some of the prior work on the "by demonstration" interaction paradigm both inside and outside the visualization community.

3.1 Interaction in Data Visualization

Interactivity can be seen as an integral part of any data visualization application. Without it, "users" become merely "viewers", and the true value of visualization diminishes. But

What is interaction?

Literature in human-computer interaction (HCI) (e.g., [18, 19, 20, 21, 22]) and data visualization (e.g., [23, 24, 3]) provide a sparse coverage of "interaction". In HCI literature, Dix et al. [25] define interaction as the *"the communication between user and the system"*. Saffer [19] defines interaction as *transaction between two entities, typically an exchange of information, but it can also be an exchange of goods or services.* In data visualization literature, Becker et al. [24] describe interaction as direct manipulation and immediate change as the two core properties. While finding a single agreed-upon definition of interaction is difficult, more specific interaction techniques can be less challenging to express and are more tangible concepts than the more nebulous concept of interaction itself [3]. Foley et al. [26] define an interaction technique as a *"way of using a physical input/output device to perform a generic task in a human-computer dialogue"*. Yi et al. [3] explain interaction techniques in data visualization as a set of tools that allow users to manipulate and interpret the data representations.

Interactive data visualizations often provide a set of features that enables users to manipulate and interpret the data representations. For example, a set of graphical elements can be added to a visual interface to enable users to filter their data based on their evolving needs. Users of interactive data visualizations often gain insight into large, complex datasets through interaction with the information, manipulating the visual representation based on their domain expertise, interactively exploring possible connections and investigating hypotheses. It is through this interactive exploration that users are able to make sense of complex datasets and gain insights.

3.2 Existing Visual Data Analysis Tools

Existing interactive data visualization tools fall into two high-level categories: *visual data analysis* and *visualization authoring* tools.

- Visual data analysis tools enable users to explore their data, generate, refine and test hypotheses, and ultimately to produce insight into their data. For example, workers in widely varying domains from finance to government to health might use tools such as Tableau [27] or Spotfire [28] to explore and gain insight about their data. Many of existing visual data analysis tools allow users to select some data attributes and then pick a *pre-defined* visual structure in which to represent the data. Such tools also enable the specification of mappings (e.g., mapping data attributes to color or size of data points). Figure 3.1 shows Tableau's main interface which is a visual data analysis tool designed for data exploration.

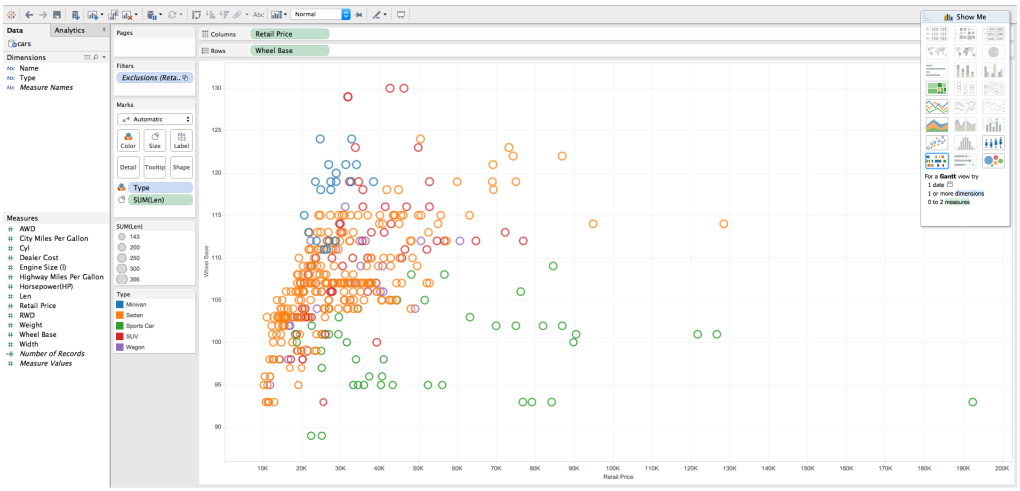


Figure 3.1: This figure shows Tableau's main interface.

- Unlike visual data analysis tools, visualization authoring tools (e.g., [29, 30, 31, 32, 33]) enable graphic designers to construct *customized* visualizations. The main focus of visualization authoring tools is more on enabling users to create customized visualizations (e.g., Infographics) rather than enabling exploratory data analysis. For example, Data Illustrator [33] enables users to draw customized visual glyphs on the canvas. It then allows the user to bind that visual glyph to the specific data attribute. Thus creating a mapping between data and customized graphical elements. See Figure 3.2 for more details.

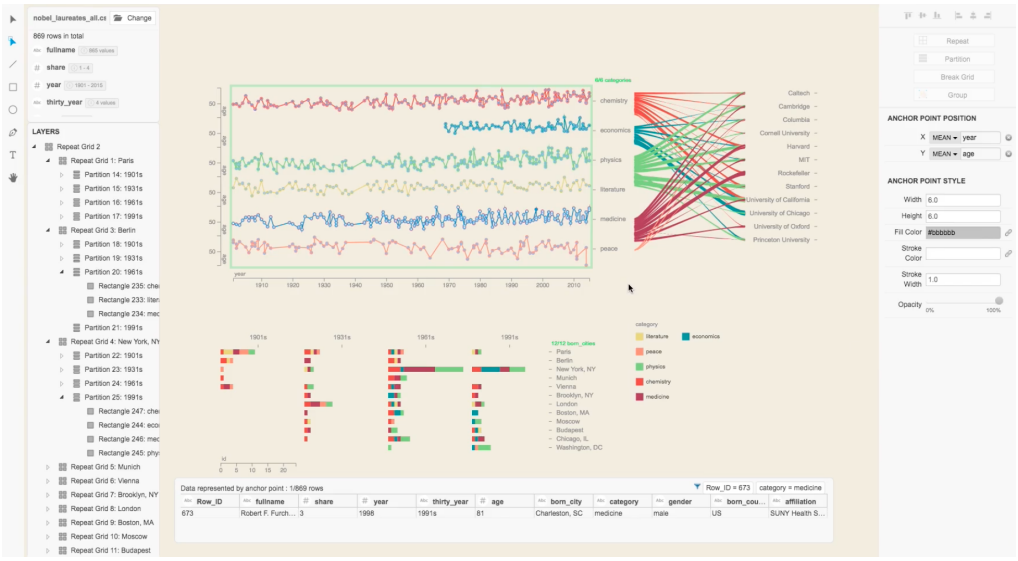


Figure 3.2: This figure shows Data Illustrator's main interface.

Our goal in this work is to mainly investigate the visualization by demonstration paradigm for visualization data analysis purposes rather than visualization authoring.

3.2.1 The Spectrum of Visual Data Analysis Tools

Visual data analysis tools range from manual to fully automated tools. On one side, we have tools such as Spotfire [28] that require users to manually specify all visualization properties by going through a set of elements on the graphical interface. On the other side, we have visualization recommender tools such as Voyager [34] that automatically suggest alternative views based on partially user-specified properties (e.g., data attribute types). Figure 3.3 shows a full spectrum of visualization tools from manual to automatic. There exist a large list of commercial and non-commercial tools that fall somewhere in this spectrum. However, we only listed some of the visualization tools for demonstration purposes.

Figure 3.3: The Spectrum of visualization tools from manual to automatic. The figure shows where each visualization tool falls in this spectrum.

Manual View Specification (MVS)

On one side of this spectrum (shown in Figure 3.3), we have tools that implement MVS. Such tools enable users to manually 1) select one of the predefined visualization techniques, 2) perform data transformation to summarize the data, and 3) assign data attributes to different visual encodings. For instance, to create a scatterplot, users must specify the visualization technique, then select data attributes to map onto the axes, and finally map any additional encodings used (e.g., size, color, etc.) to the desired attributes. In this paradigm, users are responsible for specification of all the mappings, while the system computes the resulting view. MVS frequently used in successful visualization tools such as Spotfire [28], Polaris [35], and Polestar [36].

Tools such as Tableau try to advance the visual data exploration process by automating some of the tedious and manual steps required during this process. For example, Tableau's Show Me [37] recommends appropriate visualization types in response to the characteristics of the selected data attributes (e.g., data attribute type). Tools such as Tableau can be considered more automated visualization tools compared to tools such as Spotfire [28]. However, they still implement the MVS paradigm since many of the operations still need to be performed manually. For example, in Tableau, users still need to manually assign a data attribute to size or color encodings. Thus, tools such as Tableau lean more towards manual visual data analysis tools than automated ones.

There is a fundamental difference between the tools implementing MVS and visualization by demonstration. In tools implementing MVS, users are responsible for manually specifying data mappings and the system is responsible for rendering the view based on the

specified mappings. In contrast, in visualization by demonstration systems, systems compute the most appropriate mappings and parameters based on the given demonstrations. The main difference is that visualization by demonstration aims to shift some of the burden of the manual specification from users to algorithms. While visualization by demonstration systems try to shift the specification from users to algorithms, users are still responsible for providing demonstrations manually. Due to this reason, we believe while visualization by demonstration systems are more automated than tools such as Spotfire or Tableau, they still lean toward manual tools (see Figure 3.3).

Moreover, tools implementing MVS often require users to start at an attribute level by specifying the overall mapping between attributes and visual encodings first. The main point here is that many of these tools do not allow users to manipulate individual data points, but instead deal with the full range of data by attribute. Thus, interactions in many of the existing tools happen at an attribute-level. For example, the main activity of constructing visualizations in many of the existing tools such as Tableau, Spotfire, and Polestar involves the assignment data attributes to different visual encodings (e.g., assigning a data attribute to color or size of data points in a scatterplot). In visualization by demonstration systems, users start at a low level of abstraction by manipulating the visual glyphs representing the data points to provide visual demonstrations of incremental changes. This is similar to the idea of constructive visualization [38], which is defined as *"the act of constructing a visualization by assembling blocks, that have previously been assigned a data unit through a mapping."*

Visualization Recommender Systems

Visualization recommender systems (e.g., [39, 40, 41, 34, 42]) are another category of visual data analysis tools that automatically generate a diverse set of visualizations and have the user select among them. The goal of visualization recommender systems is to encour-

age broad visual data exploration [34]. These systems aim to enable rapid and broad visual data exploration by enabling users to look at different aspects of their data. Many visualization recommendation tools have been developed to assist users to visualize their data. They suggest alternative views (visual encodings, data attributes, and data transformations) based on user-specified data of interest and computed characteristics about the data. For example, Voyager [34] is a mixed-initiative system that couples faceted browsing with visualization recommendation to support visual data exploration based on user-specified data attributes of interest (See figure 3.4). VizAssist [39] generates visualizations by requiring users to specify both desired data attributes and tasks.

Some tools that implement visualization by demonstration such as VisExemplar [6] recommend potential transformations (e.g., data mappings) based on the given demonstrations. This enables users to see a wide range of options as recommendations and pick among these options. Thus, they advocate for a broader visual data exploration by recommending views. However, the main difference between such systems and many of the existing visualization recommender systems is that they recommend views based on the demonstrations provided by users.

As we discussed earlier, visualization by demonstration systems should be considered more automated than traditional visual data analysis tools that implement MVS such as Spotfire and Polaris. However, the level of manual work required for providing visual demonstrations leans the visualization by demonstration systems towards other manual tools in the spectrum of visual data analysis tools (see Figure 3.3).

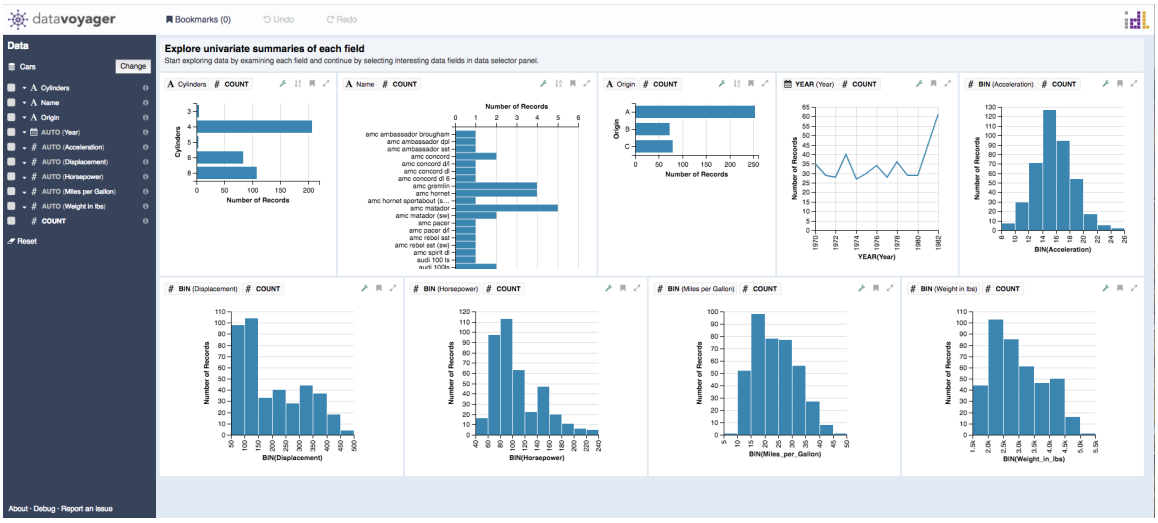


Figure 3.4: This figure shows Voyager's main interface.

3.3 A "by-demonstration" Paradigm

A "by-demonstration" paradigm has been applied to a wide range of applications in computing before coming to data visualization. In computer programming, programming by demonstration [43, 44, 45] enables users to generate code by providing visual demonstrations of some intended result. The user and the system continue to collaborate, using further demonstrations or direct edits to incrementally improve the final code. Robotics (e.g., [46, 47, 48]) has realized significant advances in domains such as manufacturing and surgical procedures by leveraging reinforcement learning and demonstration-based techniques. In doing so, domain experts (i.e., surgeons or assembly line workers) are able to demonstrate the appropriate actions to robot, instead of specifying them through low-level commands and complex parameters. Other researchers applied the "by demonstration" approach for data cleaning [49, 50]. For example, Wrangler [50] is an instance of "by demonstration" systems for data cleaning. Wrangler allows users to provide demonstrations of expected results on tabular data by directly showing results in the table view *(e.g., selecting a substring of a column to generate the transformation for creating a new column)*. Other domains that have successfully used the "by demonstration" approach include 3D drawing by demon-

stration [51], interactive database querying by demonstration [52].

3.3.1 Programming by Demonstration

The terms programming by example and programming by demonstration first appeared in software development research in the mid-1980s. In software development research, these terms initially referred to a way to define a sequence of operations without having to learn a programming language. In programming by example, the user gives an example of the expected result of the computer execution, such as a row in the desired results of a query. On the other hand, in programming by demonstration, the user performs a sequence of actions that the computer must repeat. Despite the subtle distinctions between these terms, literature often used these terms interchangeably or more generally as a demonstrational programming [43].

Pygmalion [43] is the first programming by demonstration system. Pygmalion enables programmers to create programs by editing graphical icons of the computation (e.g., loop, conditional statements). Pygmalion provides an empty canvas and provides graphical icons for the standard arithmetic, relational, and Boolean operators. Thus, programmers can directly manipulate (e.g., drag and drop) these graphical icons on the canvas, and modify them to create their program without writing any code. Pygmalion benefits programming's experience in two different ways. First, it relies on editing an artifact rather than typing statements in a programming language. Editing proven to be easy for people. Second, it eliminates an entire class of errors since operations are shown as a predefined set of graphical icons (programmers can avoid syntax errors that could appear by typing statements). See Figure 3.5 for more details about Pygmalion.

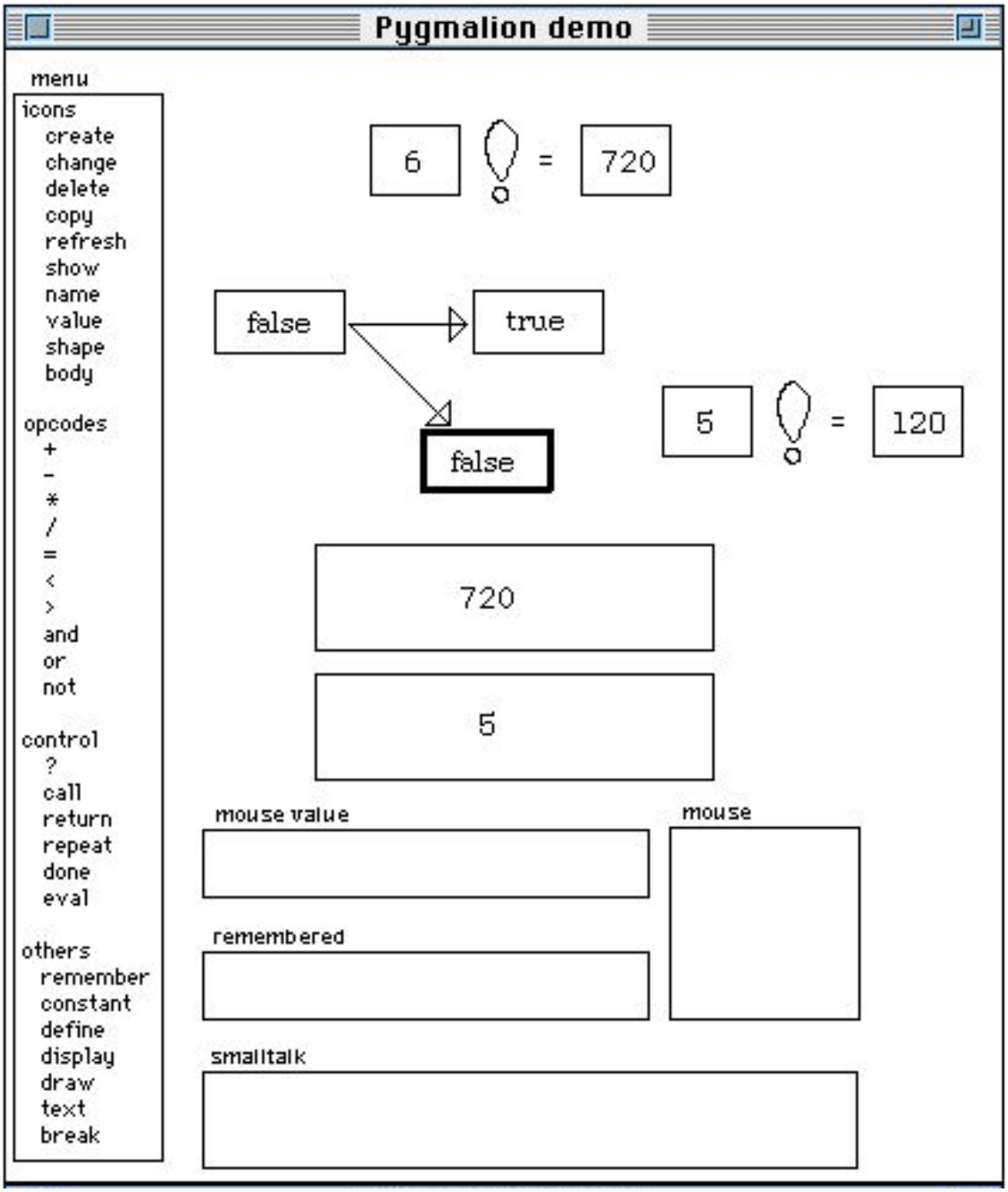


Figure 3.5: Pygmalion's interface. This figure shows a program that calculates the factorial of a given number. This program is written using Pygmalion that implements the programming by demonstration paradigm.

Peridot [53] is another example of the programming by demonstration systems. Peridot enables non-programmers to create user interfaces by demonstration. It enables users to create the graphical presentation of an interface and how the interface should respond to the

end-user mouse actions. Upon providing the demonstrations, Peridot creates reasonably efficient code that can be used with actual application programs. The programs generated by Peridot are represented as Lisp code. Peridot is using a set of condition-action rules to interference user's intention based on resulting demonstrations provided by users. Figure 3.6 shows Peridot's interface.

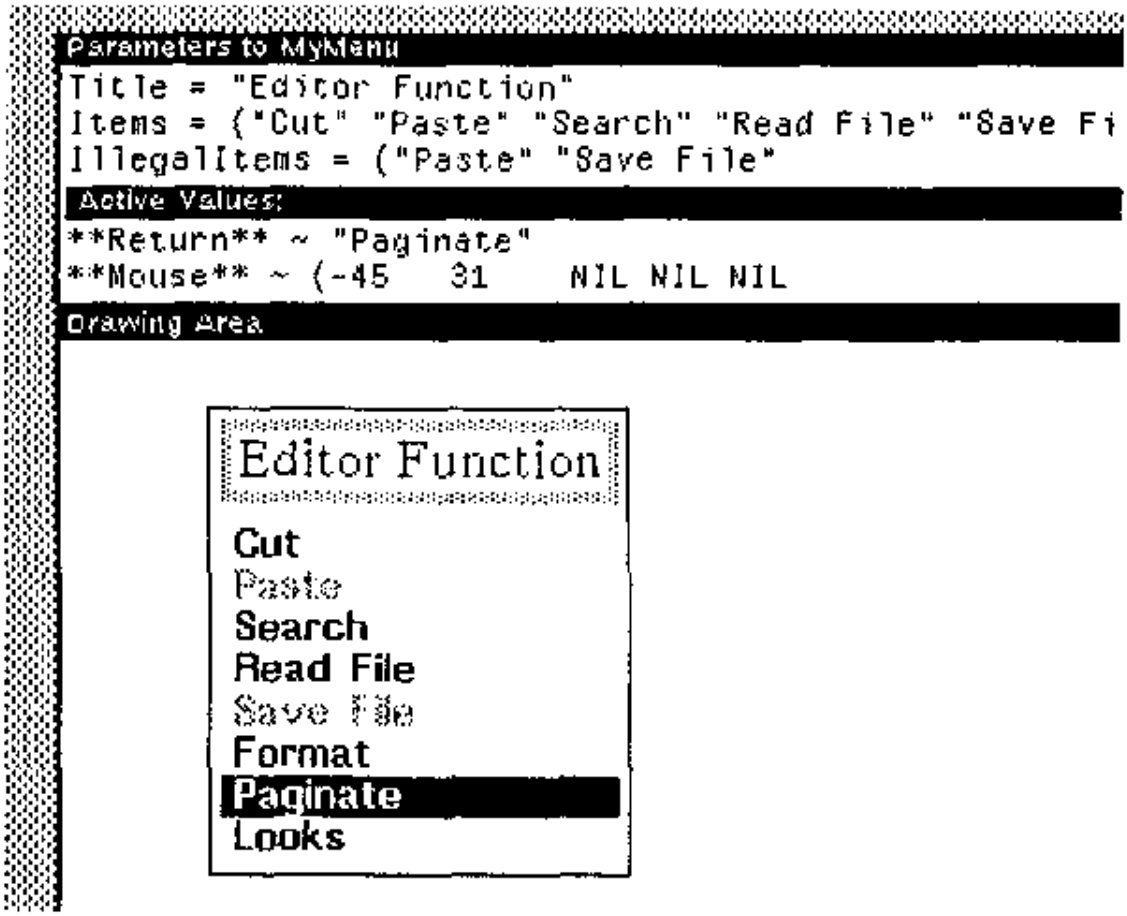


Figure 3.6: The process of creating a drop-down menu in Peridot. Users can draw on the drawing area to demonstrate their expected graphical elements. The system will generate the code for the given demonstration.

Later, Maulsby and Witten [11] developed Metamouse, an instructible agent for programming by demonstration. Metamouse helps the users of drawing tools to automate repetitive editing tasks. The main goal of Metamouse is to make programming as much like editing as possible, with minimal user interaction to clarify ambiguous situations to the system. It helps users to express their intent through a teaching metaphor [11]. Meta-

mouse enables users to demonstrate part of an action by drawing graphical constructions. In response to the given demonstrations, Metamouse calls a set of intent functions to predict the the rest of the action and carries on through further iterations. Figure 3.7 shows how Metamouse enables users to sort boxes based on their height.

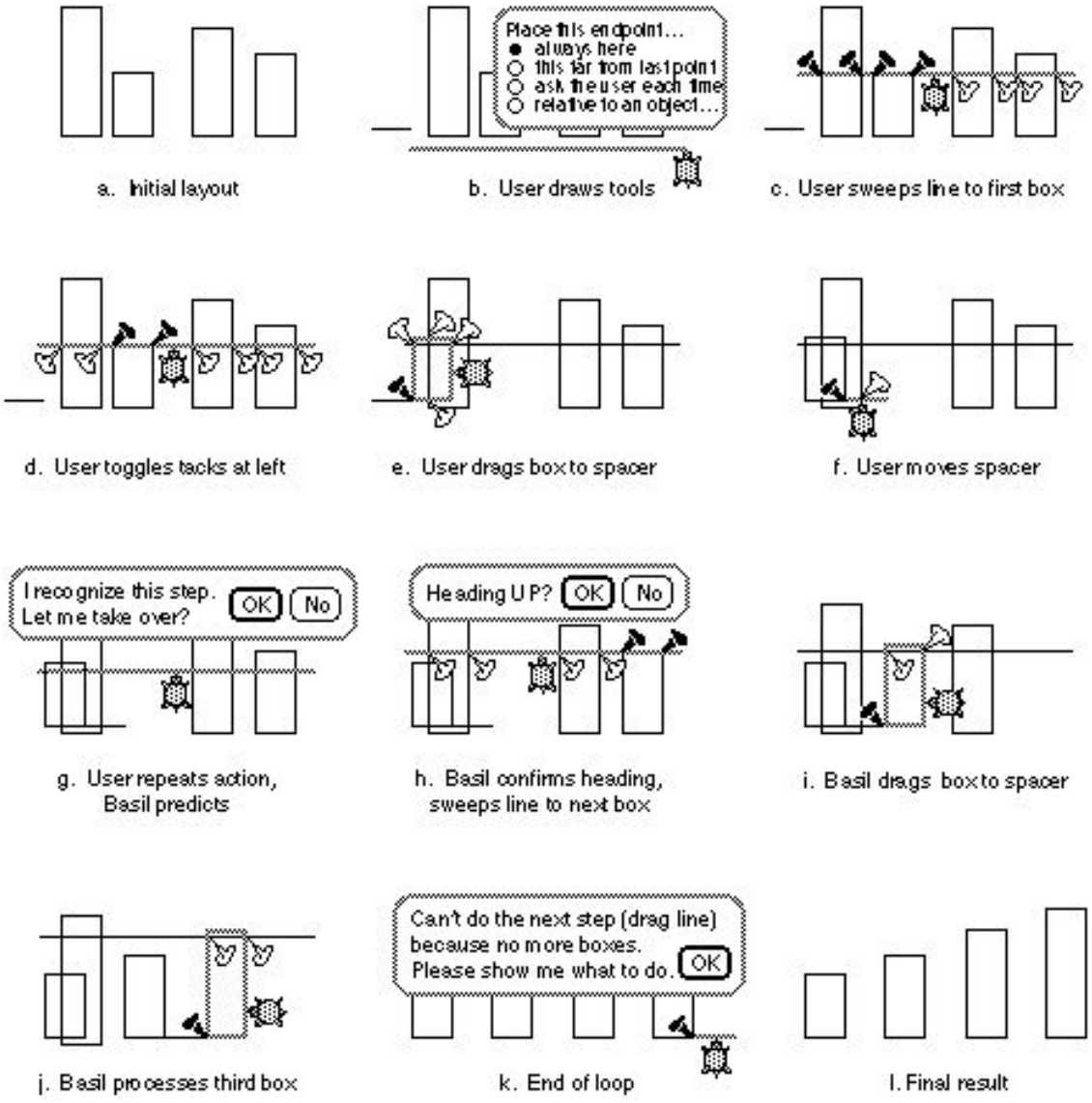


Figure 3.7: Teaching the agent to sort boxes by height.

Myers [43] developed Tourmaline, a systems that enables text formatting by demonstration. Tourmaline allows users to demonstrate their desired style on a part of the text (e.g., changing font or size of a specific word). In response to the given demonstration, the

system parse the style and apply it to the rest of text document or the table. In addition to text documents, Tourmaline enables users to change the style of the text represented in tables. See Figure 3.8 for more details.

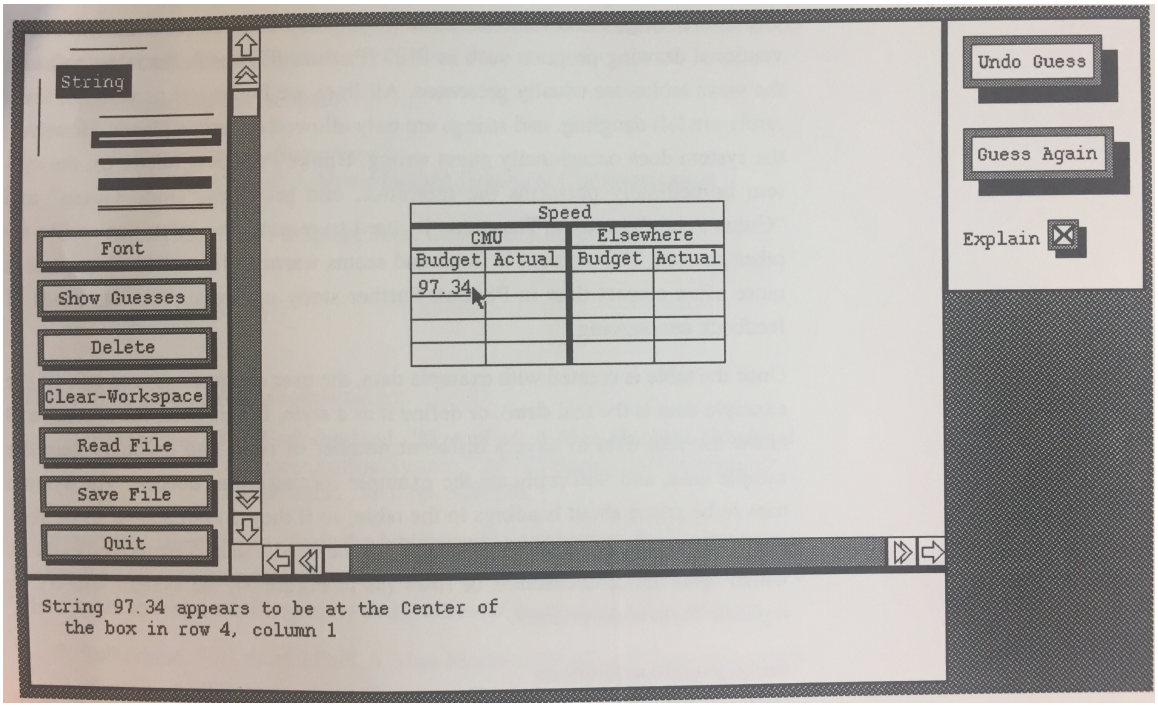


Figure 3.8: The user specifies a table by drawing an example picture. As each line and string is drawn, it snaps to an appropriate position. The window at the bottom is visible to to explain the last inference. Users can demonstrate their desired style on the text shown in each cell. The system will apply the changes to contents of all cells.

In this section, we discussed only a few of many of programming by demonstration systems [43]. These systems apply programming by demonstration to such varied problems as general purpose programming, widget building, macro creation, and text editing. Together these systems indicate a great potential and flexibility of programming by demonstration to several end-user tasks. These systems use many different and seemingly incongruent paradigms. However, they all share a common structure:

- In all these systems, users utilize some demonstrational technique to create a program. The term program refers to the internal representation that the programming by demonstration system generates. In fact, this program might never be shown to

the user. It may not be even have an explicit symbolic representation. However, it must exist.

- Once the program created, the programming by demonstration must execute the program to apply the changes which it inferred from the given demonstrations during program creation.

3.3.2 The "by-demonstration" Paradigm in Data Visualization

Providing Demonstrations by Sketching/Drawing

Myers et al. [54] proposed Gold, an interactive visualization authoring tool that enables users to draw examples of the desired customized visualizations and the system automatically generates a visualization. In response to the given demonstrations, the system uses a set of heuristics to predict the user's intended visualization and automatically renders the final visualization. See Figure 3.9 for more details about Gold's interface.

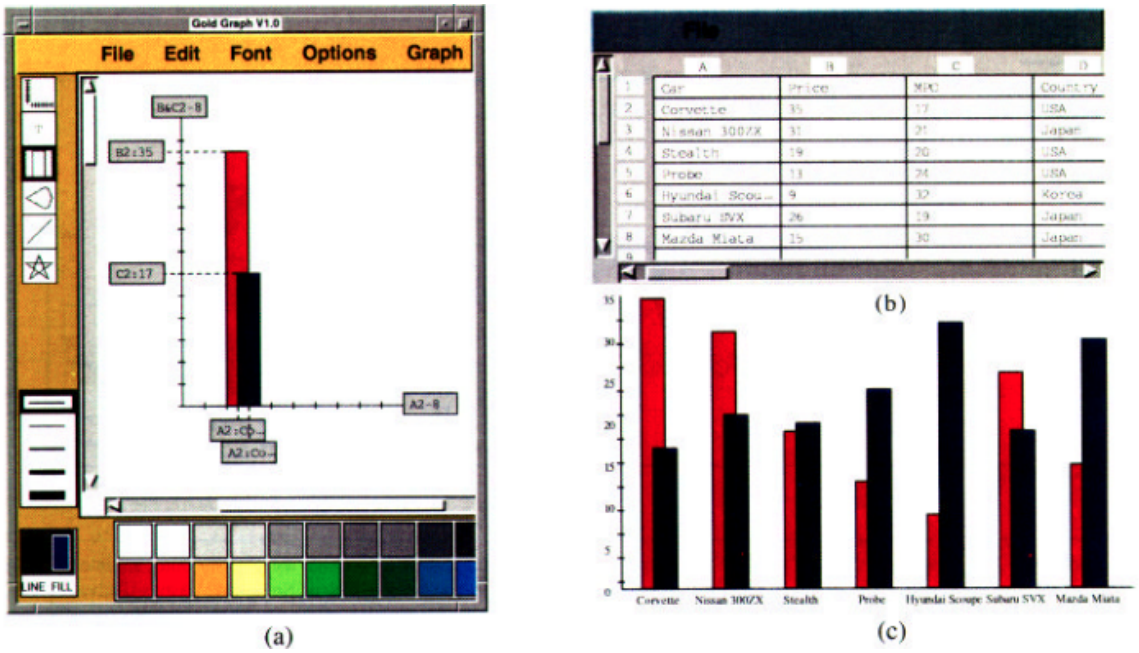


Figure 3.9: The Gold interface showing a grouped bar chart being constructed. The left panel in (a) contains a set of objects to be created, which are axes, text, rectangles(bars), pie segments, lines, and marks. Clicking on the marks brings up a pop-up menu of various graphics. The bottom and lower-left of the panel contain the line and filling style palettes. The gray rectangles are "link-boxes" that show Gold's inferences of the relationship of the bars to the spreedsheet data, shown in (b). Final visualization is shown in (c).

A related line of existing work within the visualization community has examined how sketching or drawing can be used as an approach to allow users to specify their intentions [7, 55, 56, 57]. For instance, SketchStory [7] is a system that enables users to provide demonstrations by sketching out part of the intended result (e.g., visualization's axis). From the

given demonstrations, the system then interprets user interactions and completes the visualization with new visual properties. In a different study, Schroeder et al. [55] introduced a sketching technique that enables graphic designers and artists to convey their interest in constructing multivariate time-varying visualizations by painting on a digital data canvas, sketching data glyphs, and blending together multiple layers of animated 2D graphics.

Unlike Gold [54] and the sketch-based visualization tools such as SketchStory [7] that are mainly designed for creating customized visualizations, in this proposal we are mainly interested in investigating the space of demonstration-based approach for visual data exploration. We focus on understanding how visualization tools can incorporate the visualization by demonstration paradigm to enable users to visually explore their data.

Moreover, Gold requires users to provide visual demonstration by selecting among a set of predefined objects and sketching-based tools discussed earlier rely on digital ink and ink recognition. Unlike earlier work, we are interested in exploring direct manipulation of visual representations (e.g., directly manipulating the width of bars in a bar chart) as a method for providing demonstrations. However, there are commonalities between previous work and our work in this proposal in the sense that both allow users to demonstrate their desired goal instead of manually parameterizing the visual properties.

Providing Demonstrations by Direct Manipulation of Visualization

Another line of existing work within the visualization community has examined how direct manipulation of visual representations can be used as an approach to enable users to convey their intentions to the systems [58, 59, 9]. These systems enable users to directly manipulate the graphical encodings used in visual representations as a means to convey their intent in steering the underlying models. For instance, InterAxis enables users to define and modify axes by dragging a couple of data items to either side of the x or y-axes, from which the system computes a linear combination of data attributes and binds it to

the axis [9]. Similarly, some systems allow users to adjust the distance between data items *(e.g., documents and glyphs)* to convey their interest in steering the underlying distance and similarity functions [59, 58, 60]. Each of these systems let the users perform actions on the visual objects of interest (by direct manipulation [61]) to partially convey their intended changes while interpreting the users' intentions and applying the intended changes to the entire visualization.

What really differentiates these systems from traditional direct manipulation interfaces is that they enable direct manipulation of a part of the visualization as a method for conveying changes to the entire visualization. For example, InterAxis enables users to partially demonstrate their desired axes by dragging *only* a few data points to either side of the x or y-axes. In response, the system computes how all the data points should be positioned on the axes based on the given examples. Thus, demonstrational interfaces are going beyond direct manipulation interfaces by inferring generalization from the partial demonstrations.

CHAPTER 4
PROVIDING DEMONSTRATIONS USING DIRECT MANIPULATION OF GRAPHICAL ENCODINGS

RQ2: *How do people demonstrate their intended goals using direct manipulation of graphical encodings?*

RQ3: *How effective is manipulating graphical encodings used in visualizations as a method for providing visual demonstrations?*

Andrews et al. [62] showed how space was used by their participants as both an external memory aid in which the spatial constructs carried meaning. The participants formed spatial constructs *(e.g., groups, lists, clusters)* as a means to organize the information, as well as structure their analytic process. Various studies also used spatial environments as a thinking medium to allow users to construct their visualizations incrementally. For example, Huron et al. [38] proposed a method called "Constructive Visualization", which advocates for allowing people to create visualizations by manually moving, adding, and removing physical tokens. In their study, each token represents a basic data unit. Thus, constructing a visualization means assembling these tokens to encode the data in a meaningful way (see Figure 4.1). They found people use the spatial environment to construct visualizations and explore their data, where many spatial arrangements exhibit characteristics similar to formal visualization techniques. For example, people stacked data points in the shape of bars to count the number of specific items (similar to the bar chart visualization).

Figure 4.1: Constructing a visualization with tokens. Figure from [63] used with permission.

Prior work also exists which allows users to directly adjust the position of data points (e.g., documents), interpret this feedback via a dimensionality reduction model to generate a new spatialization that better reflect the user's understanding of the high-dimensional data [59, 58, 60]. DimpVis is one of the recent systems which applies embedded interaction as a substitution to other options *(e.g., time slider)* for querying and exploring time-varying information visualizations. The system allows users to directly manipulate the data points in visualizations to perform temporal navigation of the dataset [64].

Inspired by these studies, visualization by demonstration enables users to provide visual demonstrations to a system, often by manipulating graphical encodings directly in the visualization. We refer to this as a demonstration *strategy*, as one or more manipulations may be performed to convey a user's intention in performing an operation. However, leveraging the visualization by demonstration paradigm raises several questions including *what type of demonstrations do people perform to convey their interest in performing visualization operations? how effectively do people manipulate the graphical encodings used in visualizations as a method for providing visual demonstrations? Are there demonstrations that are consistently employed to perform a given operation?* Addressing these questions through empirical studies will lead to design principles and guidelines to support further development of visualizations.

To address these questions we conduct two studies. In the first study, we conducted an experiment to understand how fast and accurate participants can manipulate 12 different

interactive graphical encodings [65]. In the second study, I conducted an elicitation study to investigate what type of direct manipulation strategies participants employ to visually demonstrate their intended changes [66].

4.1 How do people demonstrate their intended goals using direct manipulation of graphical encodings?

We conducted a qualitative study in which 10 participants each performed 15 operations on three standard visualizations (scatterplot, bar chart, and histogram). By using a think-aloud protocol and video analysis, we obtain rich qualitative data from which we extract a list of strategies people employ to perform each operation. We then identify strategies that have consensus, and strategies that are conflicting with each other. Our analysis of the results further sheds light on four high-level categories of strategies (*exemplification*, *declaration*, *instrumentation* and *selection*) from which we derive implications to help designers leverage direct manipulation of graphical encoding.

This work contributes the following to the area of interaction design in visualization: (1) a qualitative characterization of user-defined direct manipulation strategies for performing different operations on visualizations, (2) insight into users' mental models when using direct manipulation of graphical encodings as a method for user interaction, and (3) a set of actionable implications for designing interactive data visualization tools.

Because direct manipulation of graphical encodings is a recent topic of research, existing work has explored sparse points in the design space by selecting *operations* to be invoked and decided which *strategies/demonstrations* should be used to invoke each operation. Table 4.1 summarizes the operations and associated strategies/demonstrations from previous work for 2D scatterplots, bar charts, and histograms. Although we focus on visualization operations and discard analytical ones, this list gives a starting point for gathering empirical data regarding how people accomplish such operations.

4.1.1 Preliminary Studies

We conducted two preliminary studies to determine which approach to use to empirically investigate direct manipulation of encodings.

Preliminary Study 1: Paper-Based Study

In our first pilot study, we asked three participants (2 male, 1 female) to verbally explain how they would perform a series of visualization tasks using direct manipulation of graphical encodings on paper prints of visualizations. Providing participants with paper-based visualizations makes it possible to remove constraints that come with any implemented system, thus give more freedom and expressivity to participants.

We printed a bar chart representing the Cars dataset [67]. We explained the concepts of marks, encodings, labels, and direct manipulation of graphical encodings to the participants. We then explained the visualization and the data. We gave each participant three operations to perform in a random order *(e.g., How would you show that you are interested in sorting the bar chart in an ascending order?)*. We asked participants to verbalize how they would perform each operation only using direct manipulation of the encodings used in the visualization.

Overall, participants found it challenging to explain their strategies without being able to *actually* perform the operations. For example, one participant said: *"Should I imagine that I can change the width of the bar? [...] then what is the system response?"*, and another *"I have no idea what happens if I move this bar!"* Moreover, participants often did not restrict themselves to direct manipulation of graphical encodings only. For example, to sort the bar chart, one participant said he would first drag the tallest bar to the right side of the bar chart and added: *"Now I expect to see a drop-down menu that has the 'sort' option."* – thus combining direct manipulation of graphical encodings and WIMP.

Based on the results of this first study, we decided to provide an interactive tool that supports direct manipulation of different graphical encodings used in visual representations.

This would ideally help us avoid participants' confusion as to what interactions are available to them and how those interactions are implemented. In addition, we could encourage the participants to think of strategies that rely solely on direct manipulation of encodings, if no other interactions are available.

Preliminary Study 2: Partial Implementation

We conducted a second pilot study with four new participants (4 male) using a prototype with limited functionality. We developed a web-based interactive bar chart showing the Cars dataset [67] and supporting direct manipulation of graphical encodings used in bars (e.g., changing the height of a bar). We explained the visualization, data, and available interactions to the participants. Then we asked them to perform the same three operations as in the first pilot study using the system.

Participants were able to perform 8 out of 12 operations (4 participants $\times$ 3 operations) using direct manipulation of graphical encodings. Although participants found some operations challenging to invoke (e.g., assigning a new data attribute to the axis), this pilot study indicated that using an interactive prototype is an appropriate way of eliciting people's strategies for performing operations using direct manipulation of graphical encodings.

4.1.2 Study Design

In this study, we investigate *how* people perform the visualization operations listed in Table 4.1 using direct manipulation of graphical encodings. We describe the visualization and encoding types we used, the operations from Table 4.1, the dataset we used in the study and the software implementation. Then we describe our participants and settings, study procedure, and data collection method and analysis.

Visualization and Encoding Types

Given its qualitative and observational nature, we included a small number of (three) visualization types in the study to keep it within reasonable time. We selected **scatterplot**, **bar chart** and **histogram** because: i) they are among the most commonly used visualizations [2]; and ii) direct manipulation of graphical encodings has been well researched with these visualizations, as shown in Table 4.1. We studied the five encodings that have been explored in previous work for these visualizations: **position** and **color** for scatterplot and bar chart; **size** for scatterplot; **height** for bar chart; and **width** for histogram.

Datasets

Most operations listed in Table 4.1 are generic enough to not depend on the dataset being used. For example, assigning a data attribute to an axis can be performed on any scatterplot regardless of the underlying tabular dataset. Operations that involve navigating temporally, however, require a dataset that contains at least one temporal dimension. Based on this constraint, we selected the widely used cars [67] and movies [68] datasets, which have also been used in related studies (e.g., [6, 9, 64]).

Operations

We included in our study all 15 operations, 1 to 15, listed in Table 4.1. We made this choice to compare the strategies implemented in previous work for each operation with the strategies our participants would perform. We also made the choice to focus on operations for visualization construction (i.e., that are designed to specify the visualization), and excluded operations that are designed to steer underlying models used for computing visualizations (e.g., InterAxis [9]).

Table 4.1: The 15 basic operations that have been used in previous work for direct manipulation of graphical encodings in 2D scatterplot, bar chart and histogram. We use all 15 operations in our study. The last column (**Phrasing**) contains the exact sentence participants were told in our study. All the operations started with "*How would you interact with this system to show that you are interested in:* "

Encoding	**Visualization**	**Operation**	**Phrasing**
6*Position	4*Scatterplot	1 Assign a data attribute to an axis [6, 9]	Assigning the horsepower attribute to the x-axis
		2 Switch from a scatterplot to a bar chart [6]	Switching from a scatterplot to a bar chart
		3 Navigate the values of a point over time [64]	Checking if this specific car has ever had the price of 20,000?
		4 Adjust the value of a point [69]	Adjusting the value of this specific car to 30,000
gray			
black	2*Bar chart	5 Group the bars into one bar [14]	Merging the bars representing SUV and Wagon cars
		6 Sort the bar chart [6, 11]	Sorting the bar chart in an ascending order
2*Size	2*Scatterplot	7 Change the size of all points [6, 70]	Change the size of all data points, so that they are all equally bigger
		8 Assign a data attribute to the size of points [6]	Assigning city mile per gallon attribute to the size of points
5*Color	2*Scatterplot	9 Change the color of all points [6]	Changing the color of all points to red
		10 Assign a data attribute to the color of all points [6]	Assigning the cylinder attribute to the color of all points.
gray			
black	2*Bar chart	11 Change the color of all bars [6]	Changing the color of all bars to red
		12 Assign a data attribute to the color of all bars [6]	Assigning the car type attribute to the color encoding
2*Height	2*Bar chart	13 Navigate the values of a point over time [64]	Checking if the number of sedan cars have ever had the value of 35?
		14 Adjust value of a bar [69]	Adjusting the number of the SUV cars to 15
1*Width	1*Histogram	15 Expand the range of a bin in a histogram [14]	Expanding the range of this specific bin from 2009 to 2010?

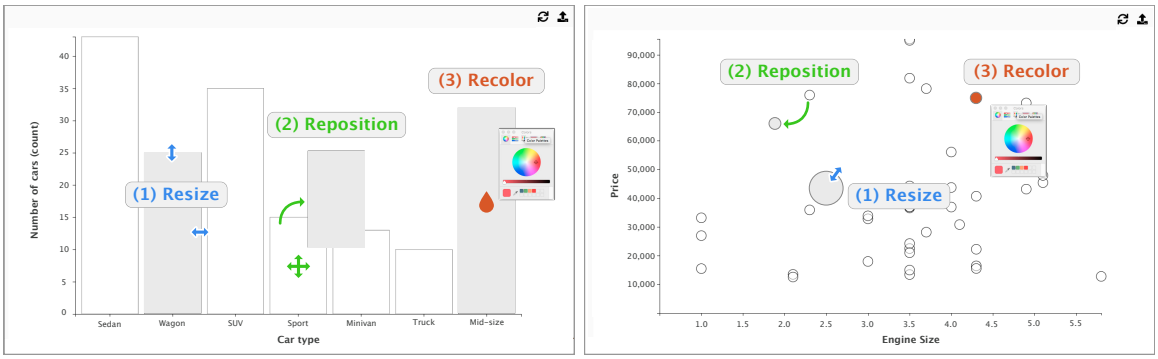


Figure 4.2: This figure shows our study platform and supported interactions for different visualizations: bar chart (left) and scatterplot (right). Users can **(1) resize**, **(2) reposition**, and **(3) recolor** bars and points directly.

Interactive Tool and Interactions

We developed an interactive visualization tool using JavaScript and the D3 library [71]. An Upload (⬆) button supports uploading the pre-defined visualizations we used in our study. A Reset button (🔄) supports resetting the visualization. We added interactivity to the encodings used in visualizations. For instance, participants could change the position, color, width, and height of the bars in a bar chart. Or, they could change the size, position, and color of circles in a scatterplot.

The tool shows one visualization at a time. Participants can directly manipulate the encodings in the visualization. For example, after the interviewer has uploaded a bar chart and asked a participant to perform an operation, the participant can use any of the provided interactions to convey their intention to perform this operation. The tool only enables participants to manipulate the encodings and does not recompute the visualization (similar to a drawing interface like Adobe Illustrator). Enabling the participants to convey their actions without the system reacting to those actions makes it possible to observe participants' unrevised behavior, and drive system design to accommodate it.

To add interactivity to the five graphical encodings (position, height, width, size, color) investigated in this study, we kept our implementation of interactions as close as possible to previous work (Figure 4.2 illustrates these interactions for the bar chart and the scatterplot):

1. Position We afforded the repositioning of circular data points in a scatterplot or bars in a bar chart or histogram anywhere on the screen through drag and drop.
2. Height and Width Clicking on a bar in a bar chart or histogram makes four small handles (black circles) appear on the four sides of the bar. Dragging the left and right handles changes the width of the bar; dragging the top and bottom handles changes its height.
3. Size Clicking on a circular data point in a scatterplot makes a small handle (black circle) appear on the perimeter of the circle. Dragging the handle changes the size (radius) of the circle.
4. Color Right clicking on a mark (circular data point in scatterplot, bar in bar chart or histogram) makes a color picker appear. Picking a color from the color picker changes the color of the mark.

Participants and Settings

We recruited 10 non-color blind participants (4 females, 6 males), aged 20–30 (mean 24.8) via email and word of mouth at our university. They were students enrolled in computer science (5), physics (1), psychology (2), and mechanical engineering (2). All reported being familiar with reading visualizations, and eight had created visualizations before. Four took the information visualization course taught at our university. Some participants had experience with visualization tools such as Microsoft Excel (8), D3.js (4) and Processing (1). None of them had participated in the pilot studies. Participants sat approximately 30–40 cm from a 13" LCD display with a resolution of 2560 $\times$ 1600 pixels equipped with a mouse and keyboard. The visualizations were shown in full screen.

Given the qualitative nature of the study, we determined participant numbers based on empirical saturation [72] – which can be reached with as low as 6 participants [73]. Two authors of this paper watched the screen-recorded videos after each session to get a sense of the strategies the participant used to perform the operations. In addition, during each

session the interviewer took notes of high-level strategies that the participant employed to perform each operation. As we progressed through our study, we discussed these notes, identifying whether the observed strategies were repeats or newly observed strategies. The sessions with participants 9 and 10 generated limited new strategies, suggesting that we had reached empirical saturation. We then discussed our informal findings as a group and decided to conclude the study.

4.1.3 Procedure

1. Introduction (10 min) Participants were briefed about the purpose of the study and their rights. After filling out the study consent form and a questionnaire on demographics and visualization expertise, they watched a three-minute video explaining the concepts of marks, encodings, labels, and axes. Participants could replay the video as they liked. Then, they were given a sample of the movies dataset printed on a sheet of paper. After the experimenter had explained to them the meanings of rows and columns, they were asked to familiarize themselves with the data for two minutes and ask any question they might have.

2. Training (10 min) In this phase, participants were shown a scatterplot, a bar chart, and a histogram all created with the movies dataset (one visualization at a time, in a random order). We explained how the system supports manipulation of different encodings used in each visualization. For instance, we showed participants that they can drag the left or right boundary of a bar to manipulate its width. We then asked participants to perform an operation on the visualization. Depending on the visualization, we asked participants to: assign a data attribute to an axis of a scatterplot; sort a bar chart in a descending order; and expand the range of a bin in a histogram. For example, the interviewer asked the participants: *"How would you interact with this system to show that you are interested in sorting the bar chart in a descending order?"* The interviewer's role in this phase was not to guide the participant, but solely to answer their questions. Such questions included: *"Am I sup-*

posed to show how I am going to do this task by manipulating these glyphs"? "Should I assume that the system is going to detect my interaction?" The interviewer did not suggest participants strategies for performing the given operations, nor gave examples or hints on how to perform an operation.

3. Main study (30 min) Participants familiarized themselves with the cars dataset like they had with the movies dataset. Then, they were shown a scatterplot, a bar chart, and a histogram created with the cars dataset, one at a time with the order randomized. For each visualization, they were asked to perform all operations associated with the visualization (see Table 4.1), one at a time in a random order. In total, each participant performed 15 operations (*8* with the scatterplot + 6 with the bar chart + 1 with the histogram). For example, the interviewer asked the participants: *"How would you interact with this system to show that you are interested in changing the color of all points to red?"* Participants were asked to perform each operation by only manipulating the graphical encodings in the visualization on the screen. Participants could also verbally explain how they would perform the operation when they could not perform it with the supported interactions. They could also suggest more than one strategy for performing each operation.

4. Wrap-up (5 min) The experimenter thanked the participants who received a $10 gift card. Participants were invited to ask additional questions about the study.

4.1.4 Data collection and Method of Analysis

We screen- and audio-recorded the whole study. During the main study, the experimenter took notes of participants' interactions. We analyzed the 298 minutes of screen-capture videos using an open coding approach [74] in three phases. In the first phase, two coders analyzed the videos together to identify the most common and unexpected patterns. For example, we observed that participants came up with many more strategies to switch the

visualization from a scatterplot to a bar chart, than for other operations.

In the second phase, both coders defined the *intended strategies* as the unit of analysis for coding the videos. An *intended strategy* is an approach that a participant explained verbally and/or performed physically to complete an operation. We included strategies that were not supported by the prototype but that participants verbally explained, e.g., *"In that case, I would select the entire screen [all data points]. Similar to how you select objects in powerpoint. Then once they are in the same group of selection, I am assuming that increasing the size of one object should increase the size of all."*

In the third phase, one coder coded the *intended strategies* for each operation. This resulted in a total of 203 annotations. Two coders then sketched out each participant's strategy and grouped these sketches into categories based on their similarities. Each sketch belongs only to one of the identified categories and all categories all exclusive. Two coders then independently named each category and counted the total number of sketches in each of the category.

4.1.5 Results

We present the strategies participants employed in our study to perform the operations using direct manipulation of graphical encodings. All but one participants could identify at least one strategy for each operation, with one participant failing to identify a strategy for one operation only. Participants suggested 4–5 strategies per operation on average. Most strategies are based on direct manipulation of the visual marks using well-known interaction patterns such as *repositioning*, *resizing* (that includes resizing the radius of points, the width of bars and the height of bars), and *recoloring* visual marks. A few strategies, however, rely on direct manipulation of other visualization elements such as axes, labels, and tick marks.

We identified 48 unique strategies across the 15 operations. Below, for each operation we present the commonalities, variety, and unexpectedness of the strategies that partici-

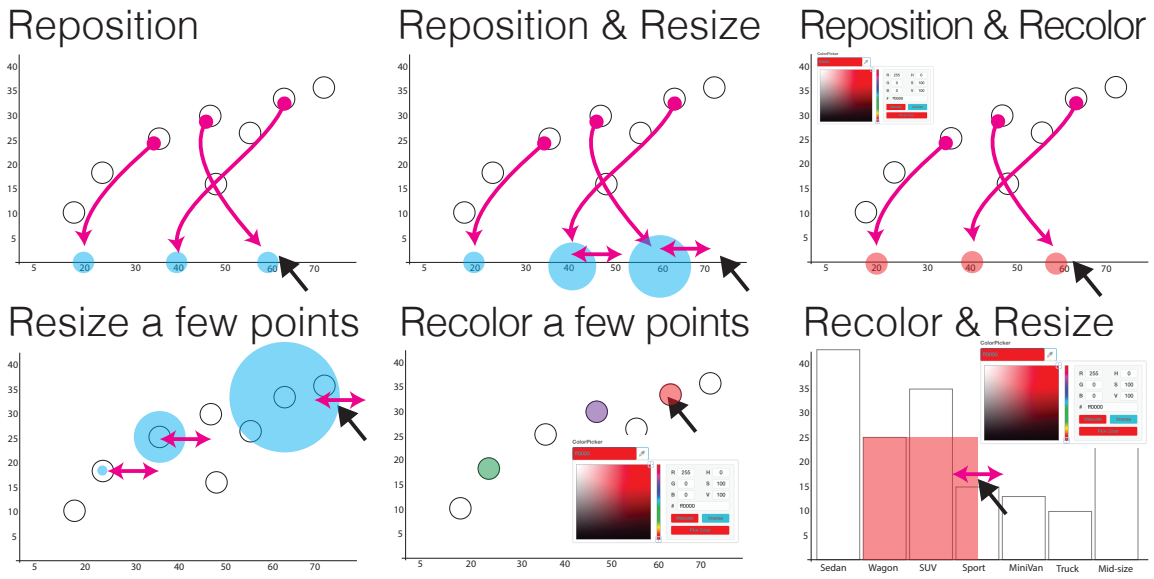


Figure 4.3: Prominent strategies to assign a data attribute to an axis (first row); and to the size or color of points in the scatterplot, and of bars in the bar chart (second row).

pants employed. We provide raw sketches drawn during the video analysis procedure in supplemental materials. For simplicity, we use "points" to refer both to the data cases in the dataset and to the visual marks representing these data points in the scatterplot (circles in our study); we use "bars" to refer to the visual marks representing data points or aggregations of data points in the bar chart and the histogram.

Assign a data attribute to an axis (1), to the size of points (8), and to the color of points (10) or bars (12) We grouped together these operations that are about assigning a data attribute to a graphical encoding. Figure 4.3 shows the main strategies used for invoking these operations.

We identified four strategies to assign a data attribute to a scatterplot axis (1). Seven used Reposition a few Points, where they rearranged a few points to sort them based on their values. This strategy has already been supported in previous work [6]. Three used Reposition & Resize, where they first rearranged a few points to sort them horizontally and then resized them based on their values. Here they manipulated the size of points to express

their intent to sort the points from lower to higher values. For example, after sorting and resizing the points, P2 said: *"So the size represents the horsepower, but at the same time the position is also horsepower. By the size [changing the size] I am trying to reinforce that the higher value should be on the right."*

One participant used Reposition inline & Recolor, where he repositioned a few data points to sort them horizontally and then colored the points based on their values to inform the system about his thought process. He stated: *"I am coloring them just to show the system they have been sorted."* Another participant also first repositioned a few data points to sort them based on the range that they fall in and then colored them (Reposition by range & recolor).

To assign a data attribute to the size of points in the scatterplot (8), 7 participants used Resize by Values, where they resized a few points differently based on their values. For example, P2 made smaller three points with a low value; then he made the point with the highest value bigger than all the other. Other participants used similar strategies: P1 resized a single random point (Resize point), and P2 and P8 first resized a few points based on their values before coloring them (Resize & Recolor). One participant also explained that he would first select all the points and then resize one of the selected points (Select & resize).

Most participants employed Recolor by Values to assign a data attribute to color, both with the scatterplot (10, 8 participants) and with the bar chart (12, 10 participants). This strategy consists of coloring a few points/bars with different values using different colors. P5 suggested Recolor & Enwidth with the bar chart, where he first colored two bars then increased the width of one of the bars to cover all the remaining bars. He explained: *"if I want to imply changes to be made to all bars, I drag the width of the bar."*

Switch from a scatterplot to a bar chart (2) Figure 4.4 shows the four most used strategies to invoke this operation out of the eight strategies we identified. Five participants used

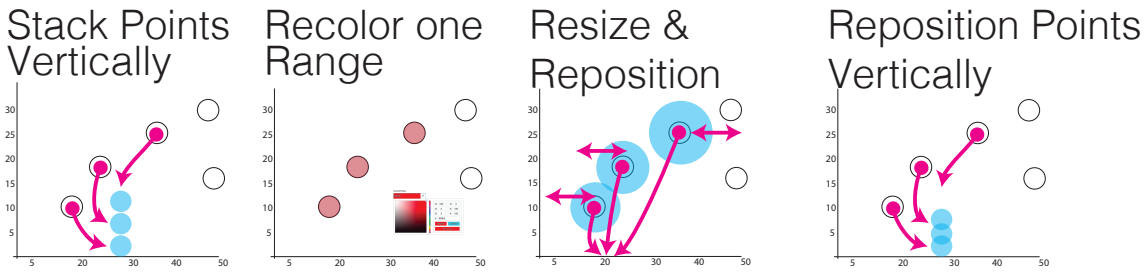


Figure 4.4: Strategies to switch from scatterplot to bar chart.

Reposition Points Vertically, where they rearranged a few points to position them vertically without any overlap. Two used Recolor one Group, where they colored a few points that fall in a specific range using the same color

Three used Resize & Reposition to Axis, where they resized a few points then moved them to the tick marks shown on the axis. When asked for clarifications, they said that they consider each resized point to be a bar where the value is mapped to the radius of the circle instead of the height of the bar. For example, P5 mentioned: *"this looks like the bar chart the bars are circular"* and P2 said: *"So basically each circle is a bar."* Two participants used Stack Points Vertically, where they rearranged a few points to position them vertically where they have some overlaps (i.e., stacking the points vertically similar to visualization by demonstration [6]).

Navigate a data point over time (3 & 13) Figure 4.5 shows the main strategies participants used when asked to check whether a data point has ever had a target value. To invoke this operation with the scatterplot (3), five used Reposition to Target, where they moved the point to the target value like in DimpVis [64]; five used Resize Point, where they resized the data point; and four used Reposition to Tick Mark, where they moved the point and dropped it on the tick mark representing the target value.

To invoke this operation with the bar chart (13), nine used Reheight to Target, where they changed the height of the bar to the target value like in DimpVis [64]; one used Recolor & Reheight, where she first changed the height of the bar to the target value and then

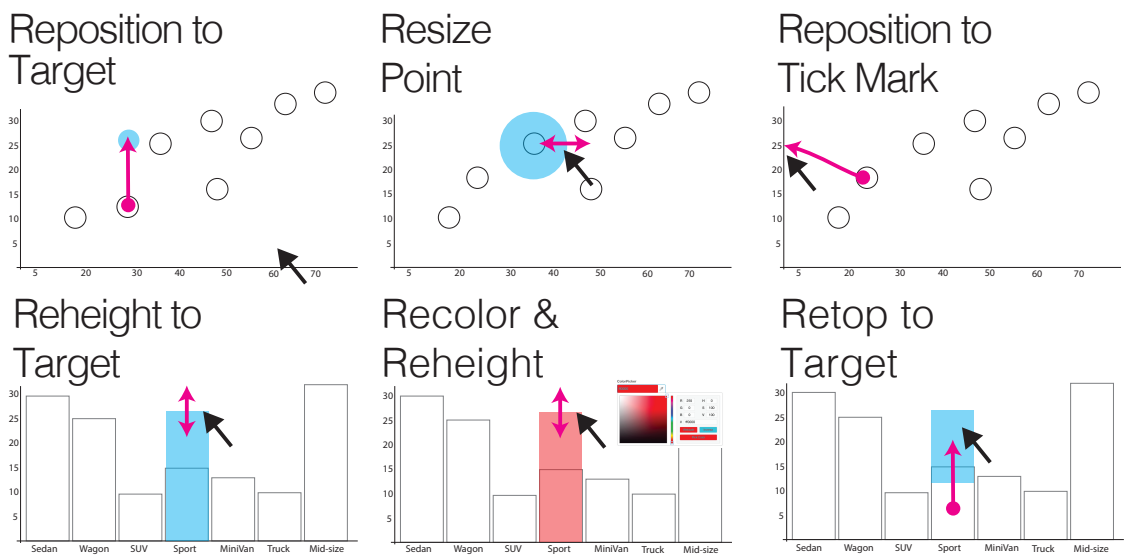


Figure 4.5: Strategies to navigate a data point over time.

colored the bar; and four used Retop to Target, where they moved the bar vertically so that the top of the bar aligns with the target value.

Adjust the value of a point (4) or a bar (14) Figure 4.6 shows the most frequent strategies for these operations. Participants used four different strategies to adjust the value of a point in the scatterplot to a target value (4). Three used Reposition to Target, where they moved the point to the target value, like in previous work [69]. Six participants extended this strategy by adding another step to it: four further colored the point (Reposition & Recolor) and two other participants resized it (Reposition & Resize).

To adjust the value of a bar in the bar chart (14), nine participants changed the height of the bar to the target value (Reheight to Target), like in previous work [69]. Two participants used Retop to Target, where they moved the bar so that the height of the bar aligns with the target value.

Group two bars into one bar (5) Figure 4.7 shows the three strategies used by multiple participants, out of the four strategies we identified. Nine used Overlay Bars, where they

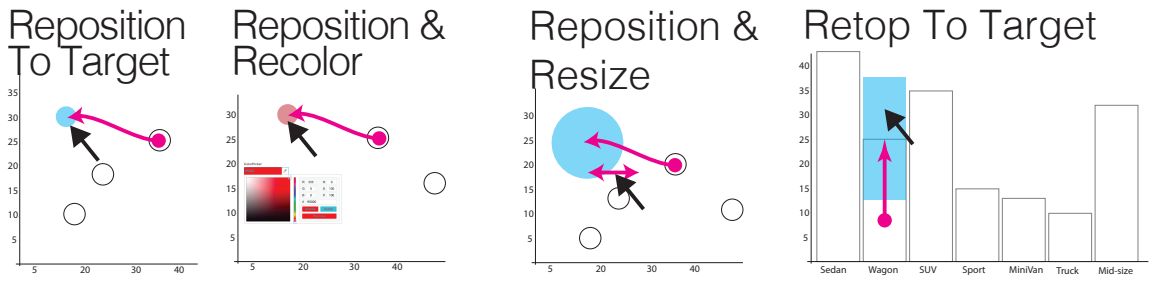


Figure 4.6: Four strategies to adjust the value of a point in a scatterplot and of a bar in a bar chart.

dragged one bar on the other one, like this has been proposed in previous work [14]. Participants found this strategy to be intuitive, e.g., *"Is there an easier way to this question? There might be others but this seems like the most intuitive one" (P1).* Two used Stack two Bars, where they stacked one bar on top of another, e.g., *"after grouping two bars, values of the two bars should add up" (P3).* Here we use "stack" to indicate the piling of visual marks on top of each other without overlap. Three participants used Recolor a few Points, where they colored the two bars using the same color.

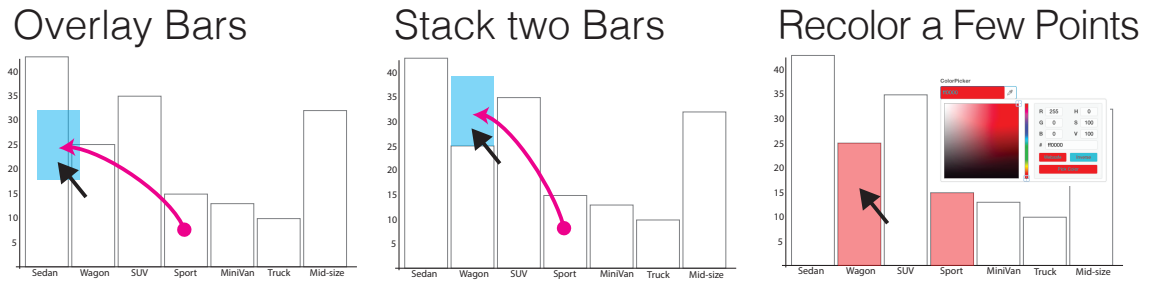


Figure 4.7: Three strategies to group two bars into one bar.

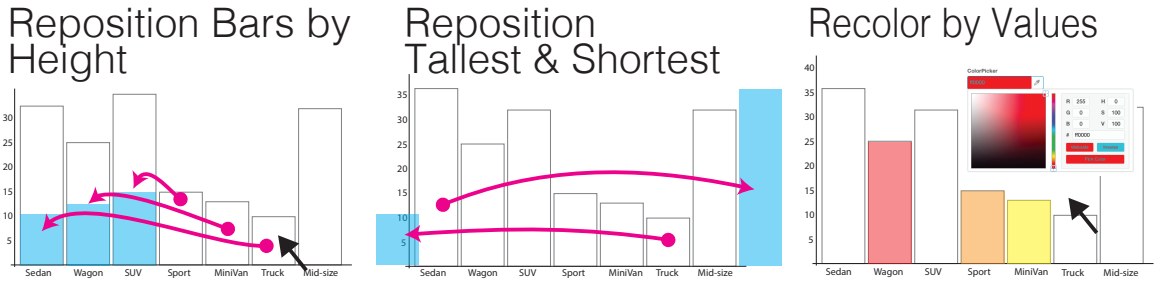


Figure 4.8: Three strategies to sort a bar chart.

Sort a bar chart (6) Figure 4.8 shows three of the five strategies participants suggested to sort a bar chart in ascending order. Nine participants used Reposition Bars by Height, where they moved a few bars based on their heights. For example, P1 stated: *"If the system is super smart, then dragging two or three bars one after another should essentially indicate that I am rearranging [sorting] them based on their values [height]."* Three participants used Reposition Tallest & Shortest, where they dragged the tallest bar to one extreme of the visualization and the shortest bar to another extreme, like in previous work [6, 11]. Some of them felt that moving both tallest and shortest bars was not necessary. For instance, P3 said: *"I pick sedan which is the biggest one and put it on here [extreme right]. This makes sense and then the system should be able to pick up what I was trying to do. And maybe if you want to make it more sure, you pick up the lowest one also put it here [extreme left]."* P6 used Recolor by Values, where she colored three different bars with ordinal colors. She explained: *"keep the first one white, then make the second one yellow. So I am assuming this is the increasing order" (P6).*

Change the size of all points (7) or the color of all points (9) / bars (11) Figure 4.9 shows the main strategies participants used when asked to execute these operations.

There were two popular strategies to change the size of all circles in the scatterplot (7). Eight participants used Resize a few Points Equally, where they resized two to four random points to make them equally bigger. For example, P2 stated: *"If I have to change the size of all of them [data points in scatterplot], then probably I should make a group of circles and they should not share the same axis, they should be very random."* Four suggested verbally Select & Resize, that consists of selecting all circles and resizing one of them.

There were four main strategies to change the color of all points in the scatterplot (9) or bars in the barchart (11). Many participants used Recolor a Few Points, where they colored two to three points/bars using the same color. Five did so to color the points in the scatterplot, and seven to color the bars in the bar chart. P1 and P6 used Recolor & Paint,

where they colored one point/bar and dragged it over a few other points/bars. Similarly, P5 used Recolor & Dip, where he first colored one point/bar, then dragged a few other points/bars such that they overlaid the colored point/bar. Finally, six participants mentioned that if selection was available they would use Select & Recolor One, where they would first select all points/bars then color one of them.

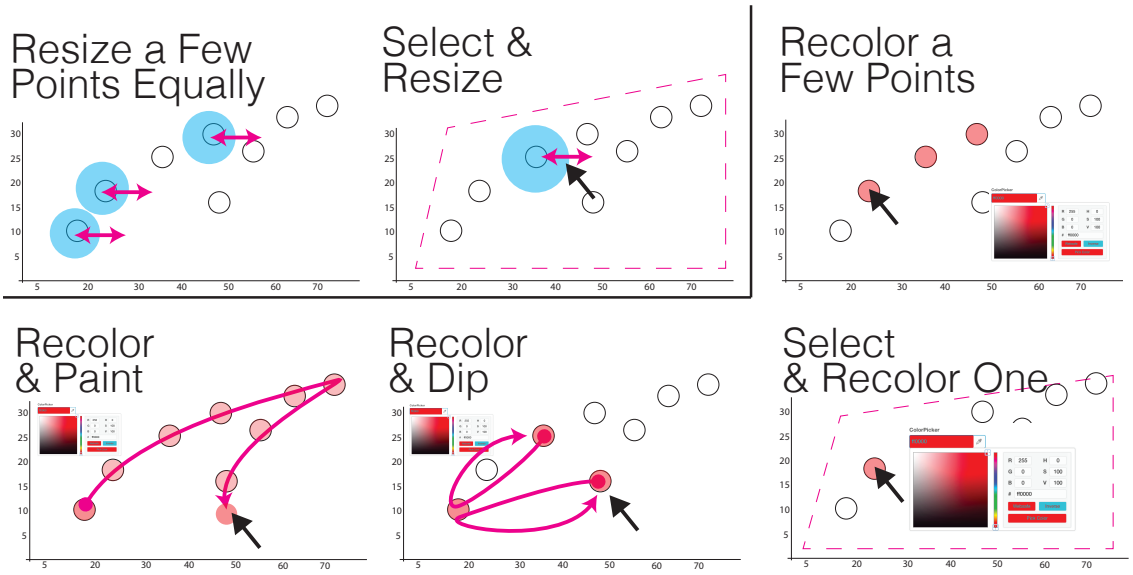


Figure 4.9: Strategies to change the size or color of all points in a scatterplot, and the color of all bars in a barchart.

Expand the range of a bin in a histogram (15) Participants suggested five different strategies to expand the range of a bin from one range to a different range (e.g., from 2008-2009 to 2008-2011). Figure 4.10 shows three of these strategies.

Nine participants used Enwidth, where they increased the width of a bar to merge it with the bars next to it, as has been proposed in previous work [14]. P7 extended this strategy into Enwidth & Recolor, where she first increased the width of the bar and then colored the bar yellow to inform the system about the latest updates: *"[...] I colored the bar to show that it has been updated" (P7).* Three participants used Overlay bars, where they dragged and dropped the bins (bars) falling in the expected target range (e.g., the bars between 2008-2011) on top of each other such that they overlaid. P2 used Recolor a few points, where he colored in the same color the bins that fell in the expected target range.

P3 used Enheight & Enwidth, where he first increased the height of the bar to exceed the x axis then increased its width to merge the bar with the bars next to it.

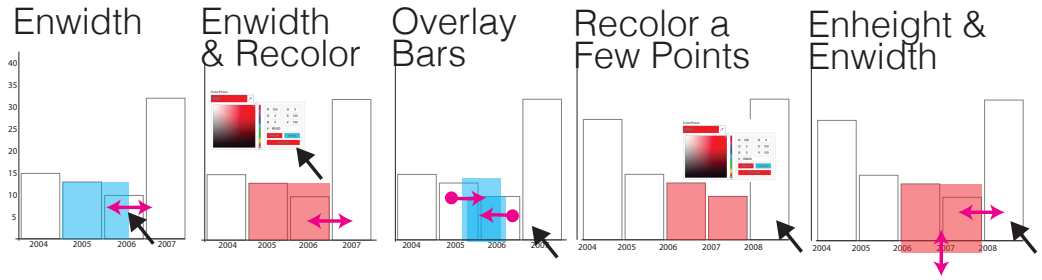


Figure 4.10: Strategies to expand the range of a bin.

4.1.6 Discussion

We structure our discussion based on Figure 4.11, that presents the 48 strategies (in columns) participants used to perform the 15 operations 1 to 15 (in rows). We first discuss the varying degrees of agreement for strategies as well as conflicting strategies (strategies that participants used to perform different operations). Then, we propose four high-level approaches for organizing the 48 strategies: *exemplification*, *declaration*, *instrumentation* and *selection*. Last, we discuss limitations of our study and future research directions.

Varying Degrees of Agreement

Some strategies were widely adopted by participants for a given operation, i.e. they have a relatively high degree of agreement. For two thirds of the operations (10/15), one strategy represents more than 50% of the distribution of all used strategies. For example, among the 12 strategies that were proposed to assign a data attribute to the color of all bars, 10 employed Recolor by Values (see 12, strategy 23). As another example, among the 15 strategies participants proposed to expand the range of a bin in a histogram, 9 employed Enwidth (see 15, strategy 41). On the other hand, some operations were performed using a wide range of strategies, with no consensus emerging. For example, to switch from a

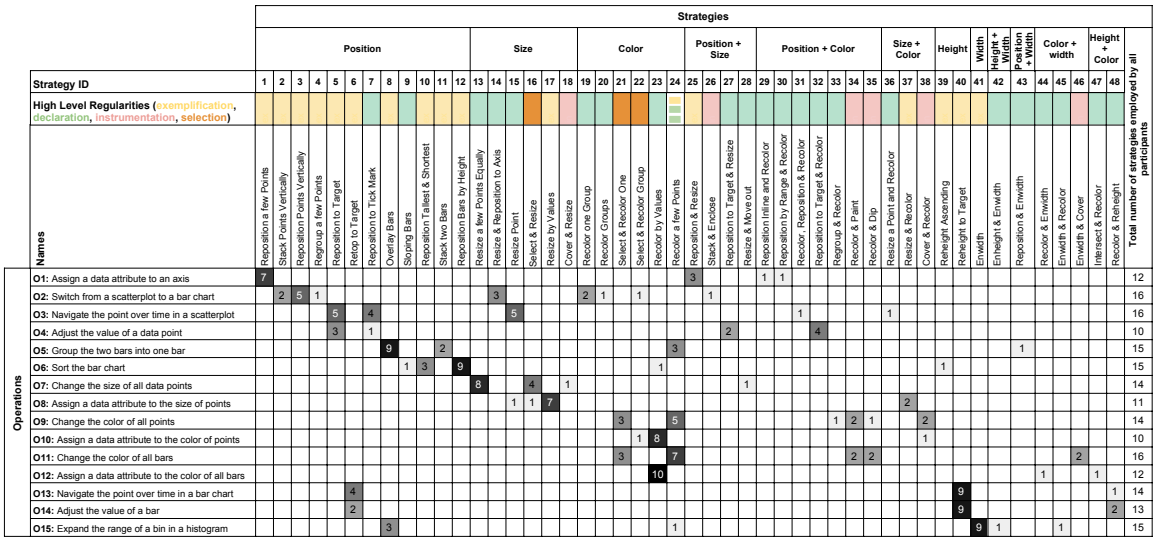


Figure 4.11: Each row is one of the 15 operations participants performed during the study, and each column is one of the 48 strategies we identified. Each cell shows the number of times participants used the strategy in column to perform the operation in row. The higher the value in a cell, the darker the background of the cell. Strategies are grouped based on the main encoding(s) involved in employing that strategy (second row in the table). For each strategy we color code the high-level approaches: exemplification, declaration, instrumentation and selection (fourth row in the table), detailed in the Discussion section. We provide detailed description of each strategy in Figure 4.12.

ID	Strategy Name	Description
1	Reposition a Few Points	Rearrange a few points to sort them based on their values
2	Stack Points Vertically	Rearrange a few points to position them vertically where they have some overlaps
3	Reposition Points Vertically	Rearrange a few points to position them vertically where they have no overlap
4	Regroup a Few Points	Rearrange a few points to put them close together
5	Reposition to Target	Move a point to the target value
6	Retop to Target	Moving a bar so the top of the bar aligns with the target value
7	Reposition to Tick Mark	Move a point to the tick mark representing the target value
8	Overlay Bars	Move one bar on the top of another bar so they overlay
9	Sloping Bars	Move bars vertically to sort them based on their heights to create ascending or descending slope
10	Reposition Tallest & Shortest	Move the tallest bar to one extreme, then the shortest bar to another extreme
11	Stack two Bars	Stack one bar on the top of another one
12	Reposition Bars by Height	Move a few bars based on their heights
13	Resize a few Points Equally	Resize a few points equally
14	Resize & Reposition to Axis	Resize a few points, then move them to the tick marks on the axis
15	Resize Point	Resize a point to the target value
16	Select & Resize	Select a set of points and resize them
17	Resize by Values	Resize a few data points differently based on their values
18	Cover & Resize	Make one point big to cover few other points
19	Recolor one Group	Color few points that fall in the same range using the same color
20	Recolor Groups	Color few points that fall in different ranges differently
21	Select & Recolor One	Select all points, then color one of them
22	Select & Recolor Group	Select a set of points and color them
23	Recolor by Values	Color a set of points based on their values
24	Recolor a Few Points	Color a few points using the same color

ID	Strategy Name	Description
25	Reposition & Resize	Rearrange a few points to sort them based on their values, then resize them
26	Stack & Enclose	Rearrange a few points to stack them vertically, then resize a circle to enclose the
27	Reposition to Target & Resize	Reposition a point to target value, then resize it
28	Resize & Move out	Resize a point, then move it out of the visualization
29	Reposition Inline and Recolor	Rearrange a few points vertically/horizontally, then color them
30	Reposition by Range & Recolor	Rearrange a few points based on a range of values, then color them
31	Recolor, Reposition & Recolor	Color a point, then move a second point to the target value and color it the same
32	Reposition to Target & Recolor	Move the point to the target value, then color it
33	Regroup & Recolor	Move a few points closer together, then color them
34	Recolor & Paint	Change the color of one data point, then move it over other points
35	Recolor & Dip	Color a point, then move other data points over it
36	Resize a Point and Recolor	Resize a single point, then color it
37	Resize & Recolor	Resize points based on their values, then color them
38	Cover & Recolor	Make one point big to cover other points, then color it
39	Reheight Ascending	Changing the height of the bar in an ascending order
40	Reheight to Target	Changing the height of the bar to the target vlaue
41	Enwidth	Increase the width of the bar
42	Enheight & Enwidth	Increase the height of the bar to exceed the x axis, then increase its width
43	Reposition & Enwidth	Drag two bars close together, then increase the width of one of the bars
44	Recolor & Enwidth	Color two bars and increase the width of one of the bars to cover all the other bars
45	Enwidth & Recolor	Increase the width of the bar, then color it
46	Enwidth & Cover	Increase the width of a bar to cover all the other bars, then color it
47	Intersect & Recolor	Drag the bottom of the bar to intersect with the tick mark, then color it
48	Recolor & Reheight	Color the bar, then change its height to the target value

Figure 4.12: Description of each of the 48 strategies participants used in our study.

scatterplot to a bar chart (O2), participants used 8 different strategies. This suggests that it might be harder to identify common strategies for certain operations.

Findings from our empirical study show that users of visualizations can effectively employ direct manipulation of graphical encoding as a means of performing operations of

varying complexity. However, the degree of agreement regarding which strategy to use varies from operation to operation. Going forward, visualization designers might consider incorporating strategies in consensus for performing operations using direct manipulation of graphical encodings. As such, our study provides a framework for collecting more empirical data and contributes to building a set of operations and corresponding consensual strategies.

Strategies in Conflict

Participants sometime employed the same strategy to perform different operations. For example, participants colored a few visual marks using the same color (Recolor A Few Points) to perform a variety of operations including: to *group two bars in one bar* (5, strategy 24); to *change the color of all points* in both the scatterplot (9, strategy 24) and the bar chart (11, strategy 24); and to *expand the range of a bin in a histogram* (15, strategy 24).

The fact that participants used the same strategy to perform different operations, in a relatively open environment, is revealing in several ways. First, it suggests that the space of strategies for performing operations using direct manipulation of encodings is not as vast as it might look. Second, it indicates that there is often no single strategy that will be unanimously used to perform an operation. Third, it shows that some strategies can be consensual for performing different operations. That is the case for Reheight to Target, for example, which was used 9 times both to navigate the point over time in a bar chart (13, strategy 40) and to adjust the value of a bar (14, strategy 40).

This many to many relationship between strategies and operations raises technical challenges in leveraging direct manipulation of graphical encoding for user interaction. Visualization designers should in the first stage avoid implementing support for strategies that different people would use for performing different operations. Future work should explore recommending to the user all possible operations in response to an employed strategy so

that the user can select the most appropriate operation (similar to VisExemplar [6]). Not only would this enable designers to support multiple operations, this could also be used to collect data on people's preferences towards developing an understanding of contextual strategies. In other words, collecting such data would enable us to analyze in which situations a given strategy is intended to trigger a particular operation.

Higher level Categorization of Strategies

By comparing the relations between strategies and operations, we identified four high-level approaches that can be used to classify the strategies for manipulating visual marks to invoke an operation (see Figure 4.11). Below we discuss these four high-level approaches: *exemplification*, *declaration*, *instrumentation* and *selection*.

Exemplification With this approach, participants modified a small number of visual marks in order to *illustrate by example* the output they were trying to achieve. Participants widely used *exemplification* to invoke operations (e.g., see Columns 1-6 in Figure 4.11). Exemplification was mostly achieved through a repetitive set of actions to show the system how a subset of the visual output should look like. For instance, one of the strategies that participants used to sort the bar chart is Reposition Bars by Height. With this strategy, they positioned the bars one by one in ascending order similar to how the bar chart should look like after invoking the operation – but they did not think that sorting manually all the bars would be necessary for the system to interpret their strategy and apply the sorting operation to the whole bar chart. The idea behind *exemplification* is similar to the visualization by *demonstration* paradigm [6]. In visualization by demonstration, one of the methods to provide visual demonstrations is to directly manipulate the graphical encodings used in the visualization to indicate a part of the expected visual output to the system.

Declaration With this approach, participants manipulated a graphical encoding different from the primary graphical encoding used in the operation. For example, although *color* is not directly linked to expanding the range of a bin in a histogram (15), one participant used Enwidth & Recolor to perform this operation. In this specific case, the participant colored the bar as a way to inform the system about the latest changes they had made. As another example, one participant used Recolor by Values to sort the bar chart. Here, while the primary graphical encodings related to the sort operation are the height and the position of bars, the participant expressed the notion of order using an ordinal color scheme.

Instrumentation With this approach, participants used a visual mark as an instrument (or tool). This resulted in relatively advanced and unexpected strategies. For example, two participants used Recolor and Paint to change the color of all points (9) and bars (11). They first colored a point/bar and then used this colored visual mark as a brush: they dragged it over other points/bars to color them. Similarly, two other participants used Recolor and Dip to perform these two same operations (9 and 11), but the other way around. They first colored a point/bar and then used this colored visual mark as a bucket: they dragged a few other points/bars such that they overlaid the colored point/bar. To turn a visual mark into an instrument, participants used the graphical encodings *color* (strategies 34 and 35) and *size* (strategies 18, 38 and 46). The idea behind instrumentation is similar to what has been proposed with constructible interfaces [75], a paradigm that is strongly grounded into instrumental interaction [76]. In constructible interfaces, one of the methods to create or modify visual marks is to turn other visual marks into instruments that can be used to perform operations.

Selection With this approach, participants expressed their interest in a selection technique (verbally because selection was not supported in the prototype). They explained that they would prefer having access to a selection option that they could use to select a subset

of points. Then they would apply an operation to a single data point rather than having to apply the same operation to multiple points. For example, participants used Select & Recolor One to change the color of all points in both the scatterplot and the bar chart.

We can further differentiate these four approaches according to the number of visual marks they require to be manipulated. Specifically, *instrumentation* and *selection* approaches were more often used by participants when they were trying to change many visual marks (e.g., 7, 9, 11) than when they were trying to change a single mark. On the other hand, *exemplification* and *declaration* were more often used to change one or two visual marks (e.g., for 3, 4, 13, 14, 15). It suggests that the number of marks to be affected by an operation is linked to the high level approach to use.

4.1.7 Limitations

This qualitative study is the first attempt to understand if and how people manipulate graphical encodings to invoke different visualization operations. As such, it cannot answer all open questions related to this problem; here we discuss the limitations of our study and findings.

Prototype Functionality Our findings must be interpreted in the context of the study prototype. To investigate how people directly manipulate graphical encodings to perform operations, we had to first design a tool that supports such functionality. Although many participants explained strategies that were not supported by the prototype, it is likely that the limited functionality of our study software has impacted their strategies. Building on our findings, future studies should consider including some of the functionality suggested by our participants, including a selection functionality.

Operation Description Although we kept the description of the operations as close as possible to previous work, the phrasing of the questions can influence the strategies employed by participants. Moreover, the nature of the operations differ from one another. For example, some operations are more direct in how they refer to the changes that need to

be made on the visual marks (i.e., the questions are congruent [77] to the task to achieve). Despite this limitation, we observed a multiplicity of strategies for all operations.

4.2 How effective is manipulating graphical encodings used in visualizations as a method for providing visual demonstrations?

To investigate how fast and accurate users manipulate different graphical encodings, we conducted a within-subjects study in which participants performed magnitude production tasks *(e.g., change the value of the interactive graphical encoding to x% of its current value)* [65]. In an attempt to support more familiar and natural methods of user interaction, we chose to run the study as an online experiment so participants could use the setups and environments familiar to them *(e.g., their own machines with their own familiar input configuration)*.

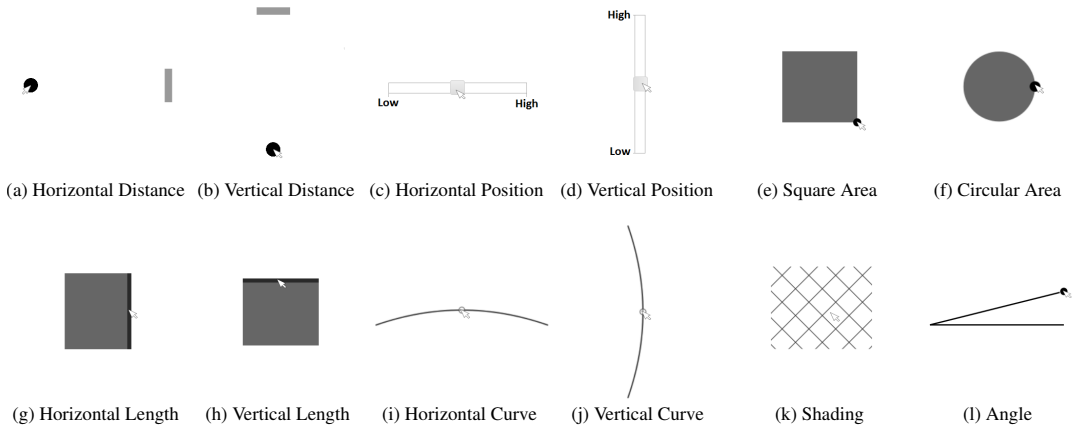


Figure 4.13: The 12 interactive graphical encodings assessed in this study, designed based on seven common elementary graphical encodings used in data visualization: *distance, position, length, angle, curvature, shading*, and *area*. Interactive graphical encodings are elementary graphical encodings that can be directly manipulated or adjusted.

4.2.1 Interactive Graphical Encodings

To study interactive graphical encodings, we first selected seven common elementary graphical encodings (following previous work [78, 79]) used to construct many visualizations today: *distance, position, length, angle, curvature, shading*, and *area*. We then developed 12 interactive versions of these graphical encodings by taking horizontal and vertical orientations into account for *distance, position, length* and *curvature*; see Figure 4.13. This

section describes the types of interactive graphical encodings used in the experiment.

Distance (Horizontal and Vertical). This interactive graphical encoding contains a rectangle (a reference position) and a small circle as the controller (see Figures 5.12-a and 4.13-b). Participants could adjust the *distance* between the circle and the reference rectangle by dragging the circle with a mouse along a single dimension. This encoding is common in visualization systems that allow users to adjust the distance between visual elements where similar elements are spatially close to one another *(e.g., [58])*. For our analysis, we calculated the error of participants' responses by comparing the distance (in pixels) in the user's response to the expected response.

Position (Horizontal and Vertical). This interactive graphical encoding presents a horizontal or vertical slider to the participants (see Figures 4.13-c and 4.13-d). Variations of sliders are commonly used for filtering in different visualization systems. To interact, participants moved the *position* of the box at the center of the slider by dragging it with a mouse. While the *Position* and *distance* encodings are similar, we note a key difference between the two: the *position* encoding presents users with explicit low and high points, and it includes a visible one-dimensional scale in the background (the slider's scale). The primary reason for including both encodings was to see whether adding an explicit movement boundary (low and high points along with the background scale) affects user performance. For our analysis, to compute the error of participants' responses, we compared the user's position of the slider box on the scale versus the expected position.

Area (Rectangular and Circular). This interactive graphical encoding came in two variations: square and circle. Participants adjusted the *area* of the shape by dragging a small handle (tiny black circle) on the perimeter of the object; see Figures 4.13-e and 4.13-f. One of the applications of *area* manipulation is *rectangular brushing*, in which users se-

lect a subset of the data items by drawing a rectangle with an input device (examples can be found in the D3.js visualization library [80]). For our analysis, we compared the area of the user's object versus the expected area to compute the error of participants' responses.

Length (Horizontal and Vertical). This interactive graphical encoding involves re-sizing the *length* of a rectangle (see Figures 4.13-g and 4.13-h). Participants adjusted the *length* by dragging the right or top edge of the rectangle with a mouse cursor. Directly manipulating the length of a bar has been used as a method for filtering data *(e.g., [9])*. For our analysis, we compared the horizontal length (or height) of the rectangle (in pixels) versus the expected length to compute the error of participants' responses.

Curvature (Horizontal and Vertical). The implementation of this interactive graphical encoding is comprised of a curved line with a small circular handle at its center. Participants adjusted the *curvature* of the line by dragging the handle along a single dimension (horizontally or vertically); see Figures 4.13-i and 4.13-j. For our analysis, we compared the horizontal or vertical distance (in pixels) between the circle and the line segment between the end points of the curve versus the expected distance to compute error. Similar to Cleveland and McGill's experiments [81], we used the horizontal or vertical distance between the circle (mid-point of the curve) and the line segment connecting the end points of the curve as our measurement metric. For our analysis, we compared this value to the expected distance to compute error.

Shading. This interactive graphical encoding contains a rectangular area with cross-hatched shading (see Figure 4.13-k). Participants adjusted the *density* of the hatch pattern by dragging the mouse cursor up or down. This interaction was selected for consistency with the other interactive graphical encodings. *Shading* is often similar to color saturation for graphical perception [82], and these encodings are commonly used in many different types

of visualizations, including infographics, choropleths, and heatmaps. For our analysis, we compared the number of cross-hatched rectangles in the object versus the expected number of cross-hatched rectangles in the object to compute the error in participants' responses.

Angle. This interactive graphical encoding contains two line segments that meet at an angle with a handle (a small black circle) at the end of one of the line segments (see Figure 4.13-l). Participants could adjust the inner *angle* between two lines by dragging the handle with a mouse. Angular representations are common in pie charts, and interactive angles could also be used in other forms of visualizations, as graphical perception of static angles has been shown to be fairly accurate [78, 79]. For our analysis of interaction accuracy with the angular encodings, we compared the inner angle (in degrees) between the line segments versus the expected inner angles to compute the error of responses.

4.2.2 Hypotheses

Based on earlier work [83, 78, 79] and our own experiences, we considered the following hypotheses for our study:

- **H1:** We expected accuracy and interaction time to be different among different interactive graphical encodings. More specifically, we expected accuracy to be better and interaction time to be faster for *distance, position*, and *length* compared to *area* and *shading*. Prior research shows people can perceive *length* and *position* more accurately than *area, curvature*, and *shading* in static visualizations [78, 79]. We also expected that *curve* and *angle* would fall somewhere in the middle of the ranking for both accuracy and interaction time.

- **H2:** We hypothesized that accuracy of horizontal interactive graphical encodings would be higher than for vertical orientations. Research by Benner [83] found that humans are better at estimating *position, distance*, and *length* of objects that are oriented horizontally, as compared to those with vertical orientation. Thus, we decided

to include both horizontal and vertical orientation for each interactive graphical encoding when applicable for the graphical encoding type (that is, some types did not have natural horizontal and vertical variations).

4.2.3 Participants

The study was conducted online by invitation to students at a single university. Of the 46 participants who began the study, 35 completed the study (22 male, 13 female). Ages ranged from $18-34$ years. Participants were mostly undergraduate and graduate students in science and engineering programs, and they were familiar with plots and computers. The participants were provided with the URL and could participate in the study using any device. Participants who completed the study were compensated with a $5 Starbucks gift card. In addition, the three participants with the most accurate and fastest responses were given a $25 gift card.

We also collected logs containing users' operating systems and input devices. Participants used different operating systems (20 Mac OS, 11 Windows, and 4 Linux users) to participate in our experiment. Moreover, 18 of the participants used a mouse and the rest used a trackpad to adjust the interactive graphical encodings.

4.2.4 Task

Each interactive graphical encoding was accompanied by instructions that required the participant to adjust the interactive graphical encoding to a target value. A target value is a certain percentage that we asked each participant to adjust the interactive encoding to. For example, for the *length* encoding, we asked participants to adjust the length to 150% of its current value. Participants could adjust the graphical encodings' values by directly manipulating them, as described previously.

In a pilot study, participants reported sometimes losing track of the starting value for the question while performing a task. To address this feedback, we made sure the interface

for the experiment always showed the initial value as a reference point while users interacted with encodings. Since the order of encodings and target values was randomized, this reference point helped users to keep track of the initial position for the given encoding. The initial value was shown as a semi-transparent reference point for all the graphical encodings except shading. For shading, we showed two shadings side by side, where the right side always showed the initial value, and the the left side was the one that the participants could interact with.

Our task resembles a magnitude production task [**wagner2006geometries**]. This task is motivated by the fact that while users manipulate a visual element on the interface *(e.g., position of a knob on a slider)* they constantly compare its current value to a reference point [84]. In our study, the reference point is the reference value *(i.e., the starting value encoded)*.

4.2.5 Training Procedure

At the beginning of the study, participants were briefed about the purpose of the study and their rights. They then were instructed how to complete the experiment.

In order to familiarize the participants with the graphical encodings, interactions, and questions, participants first completed 12 practice trials (one trial per interactive graphical encoding). Each trial included the task description *(e.g., make the inner angle between the two lines 200% of its current value.)* and the interaction instructions *(e.g., drag the black circle to move the line)*. To provide feedback after completing each trial, participants were shown a visual comparison between their response and the correct answer for each trial. Thus, the task description and training showed the participants how to perceive and manipulate each encoding.

4.2.6 Experimental Procedure

Participants performed seven trials for each of the 12 versions of interactive graphical encodings, and each trial had a different target value (25%, 50%, 75%, 125%, 150%, 175%, and 200%). Participants performed 84 tasks (12 interactive graphical encodings $\times$ 7 trials) with randomized task order. Current value (starting point) of all interactive graphical encodings was 100%.

After completing the practice trials, participants began the main experiment with the 84 randomized trials. For each question, we logged interaction time and the changes in accuracy made every millisecond. Interaction time started as soon as participants started interacting with an interactive graphical encoding.

4.2.7 Task Performance Results

In this section, we first describe the methods used to analyze the data collected from the experiment. We then provide an overview of our results, with more detailed quantitative results listed in Figure 4.14. The collected data has 2940 answers (84 trials $\times$ 35 participants). We measured both interaction time and accuracy for each trial. Interaction time was measured by computing the total time each participant spent interacting with a primitive. Accuracy percentage was measured by subtracting the percentage of response error from 100, where the response error is:

$$Error = \frac{|ResponseValue - ExpectedValue|}{ExpectedValue} \times 100$$

To account for data quality from online data collection, outlier handling was performed to account for trials where participants were likely to have disruptions or mistakes that were greater than would be expected with a usual attempt. For instance, trials having very long completion times were excluded because users likely did not spend the entire duration performing the single task in such cases. We excluded 268 (9%) of the collected

responses as outliers based on interquartile range (IQR), where an outcome was considered an outlier if it was more than 1.5 times the size of the IQR away from either the lower or upper quartiles. The outlier distribution of the 9% of trials was spread across encoding type (2.1% shading, 1.9% area, 1.6% curvature, 1.1% length, 0.9% position, 0.7% distance, and 0.7% angle). To some extent, more outliers were associated with encodings with lower performance, but the variation was not extreme. We applied the outlier removal procedure for each encoding separately.

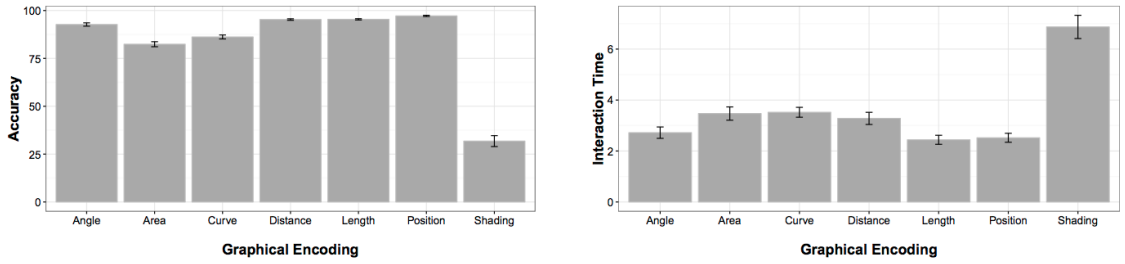
4.2.8 Data Analysis

To address our first two hypotheses, we needed to test how the different interactive graphical encodings (**H1**) and differences in adjustment orientation (horizontal or vertical, as described in **H2**) affected the performance outcomes of interaction time and interaction accuracy. We provide all relevant materials for this study online [1]: software for running the experiment, anonymized results, and statistical test results.

To analyze the differences among the various interactive graphical encodings, we first calculated separate mean performance values for all trials. That is, for each participant, we averaged outcome values of trials for each interactive graphical encodings. To test effects due to orientation, performance outcomes for each level (horizontal and vertical) were averaged for the trials of each interactive graphical encoding with the appropriate orientation. Adjustment orientation was only varied for four graphical encoding types (*distance*, *position*, *length*, and *curve*).

To test the combined effects of interactive graphical encodings and adjustment orientation, we would ideally turn to a two-way factorial analysis of variance (ANOVA). However, because adjustment orientation was only variable for a subset of the graphical encodings, a factorial analysis was not appropriate for the unbalanced design. As an alternative, we conducted a one-way repeated-measures ANOVA to test for differences among the various

[1] `http://va.gatech.edu/encodings/`



Test of Within-Subjects Effects
Interactive Graphical Encodings ($F_{(1.7,58.5)} = 401.5, p < 0.001, \eta_p^2 = 0.92$)

Pairwise Comparisons (ranked from most accurate to least)
***p* values are corrected using Bonferroni correction.**

Graphical Encodings		
Position	**vs.** Angle, Area, length, Curve, Distance & Shading	($p < .01$)
Length	**vs.** Angle, Area, Curve and Shading	($p < .01$)
Distance	**vs.** Angle, Area, Curve & Shading	($p < .01$)
Angle	**vs.** Area, Curve & Shading	($p < .01$)
Curve	**vs.** Shading	($p < .01$)
Area	**vs.** Shading	($p < .01$)

(a)

Test of Within-Subjects Effects
Interactive Graphical Encodings ($F_{(2.9,100.0)} = 95.2, p < 0.001, \eta_p^2 = 0.73$)

Pairwise Comparisons (ranked from fastest to slowest)
***p* values are corrected using Bonferroni correction.**

Graphical Encodings		
Length	**vs.** Area, Curve, Distance & Shading	($p < .01$)
Position	**vs.** Area, Curve, Distance & Shading	($p < .01$)
Angle	**vs.** Area, Curve, Distance & Shading	($p < .01$)
Distance	**vs.** Shading	($p < .01$)
Curve	**vs.** Shading	($p < .01$)
Area	**vs.** Shading	($p < .01$)

(b)

Figure 4.14: Performance results for different interactive graphical encodings along with statistical test results. Mean accuracy is shown in (a), and mean interaction time is shown in (b). Error bars represent standard error.

interactive graphical encodings, and a separate two-way repeated-measures ANOVA to test for interactions between interactive graphical encodings and adjustment orientation for the subset of encodings that had horizontal and vertical versions.

Before testing, we checked that the collected data met the assumptions of appropriate statistical tests. The assumption of normality was satisfied for parametric testing, but Mauchly's Test of Sphericity indicated that the assumption of sphericity had been violated for both accuracy and speed. To address this issue, we report test results with corrected degrees of freedom using Greenhouse-Geisser estimates for $\epsilon < 0.75$ and otherwise with Huynh-Feldt correction.

4.2.9 Findings

In this section, we organize the results of the statistical tests by independent variables and interactions.

Interactive Graphical Encodings. We found significant main effects for both accuracy and time for encodings, and we followed up with Bonferroni-corrected posthoc com-

parisons; see Figure 4.14.

Figure 4.14-a shows accuracy by interactive graphical encoding type. *Position* has the best and *shading* has the worst accuracy. Accuracy of *position* was significantly better than all other interactive graphical encodings. However, Figure 4.14-a shows that practical advantages are notably small for *position* over *length* and *distance*, even though standardized effect sizes are high (Cohen's $d = 0.84$ between *position* and *length*, and $d = 0.91$ between *position* and *distance*). Pairwise comparisons did not detect significant differences among *length, distance*, and *angle*. In other words, *length, distance* and *angle* were interpreted with similar accuracy. We also found that *shading* was significantly less accurate than all other encodings. Moreover, *area* and *curve* fall somewhere in the middle in terms of the accuracy ranking.

Participants had the fastest interaction times using *length*, *position*, and *angle*, respectively. Although results of pairwise comparisons did not show significant difference among the three interactive graphical encodings, they were significantly faster than *area, curve, distance*, and *shading*. *Curve, distance*, and *area* were in the middle in terms of time. Results indicate that ranking of the interactive graphical encodings by accuracy is slightly different from the ranking based on interaction time. Rankings of the encodings for both accuracy and interaction time are shown in Table 4.2. *Position*, *length* and *angle* are among the best and *shading* is the worst in term of both accuracy and interaction time. More details are shown in Figure 4.14.

Adjustment Orientation. The tests failed to detect significant main effects of adjustment orientation for either accuracy ($F_{(1,34)} = 0.7, p > 0.05$) or interaction time ($F_{(1,34)} = 6.6, p > 0.05$); therefore, the results do not serve as evidence for interaction performance being influenced by horizontal or vertical orientation.

Interactive Graphical Encodings $\times$ Adjustment Orientation. There was a significant interaction between graphical encodings and adjustment orientation for both accuracy ($F_{(1.7,58.5)} = 4.7, p < 0.5$) and interaction time ($F_{(2.5,87.6)} = 17, p < 0.05$). While partici-

pants had more accurate interactions for the vertical versions of *length*, *curve*, and *position*, accuracy was lower for the vertical *distance*. In terms of time, participants were faster with vertical *position* and *distance* than the horizontal versions. This was opposite for *length* and *curve*; participants had a slower interaction with vertical *length* and *curve* than their horizontal versions.

Table 4.2 shows rankings of the interactive graphical encodings based on the different metrics assessed alongside rankings of graphical encodings provided by Cleveland and McGill [78]. In each column, interactive graphical encodings are ranked from best to worse according to performance in each metric. For example, *position* has the best and *shading* has the worst accuracy in our study. Unlike the study by Cleveland and McGill [78], we did not include some graphical encodings such as *volume*, *color* and *direction*. Using *volume* is not recommended in many visualizations due to confusion that this type of graphical encoding might cause [85]. Similar to previous work [79], we excluded *color* mainly because we lacked control over participants' display configurations in the online study.

In our ranking, accuracy of *curve* was not significantly different from *area*. Note that this was a different result as the ranking provided in previous work (see Table 4.2), which found *area* to be more accurate than *curvature*. While average accuracy of *curve* was higher than *area* in our ranking, the pairwise comparison did not indicate a significant difference between their accuracy. Additional testing would be required to determine the ordering or equivalence between these two encodings. As previous work [79] discusses, the study by Cleveland and McGill did not find a significant difference between *length* and *angle* encodings (as psychophysical theory would predict [78, 1]). However, the results of our study found a significant difference between these two encodings in terms of accuracy.

4.2.10 Discussion

Designers might find ranking of one metric more important than another depending on their requirements. As an example, one might argue that the accuracy of an interactive

Table 4.2: Ranking of the interactive graphical encodings based on completion accuracy and interaction time. Rows indicate significant differences between encodings.

Our Study		**Cleveland & McGill [78]**
Time	Accuracy	Accuracy
Length, Position, **Angle**	Position	Position
Distance, **Curve**, **Area**	Length, Distance	Length, Direction, Angle
Shading	**Angle**	Area
	Curve, **Area**	Curve, Volume
	Shading	Shading, Color

graphical encoding plays a more important role than interaction time. Depending on the application of the visualization, designers might take into account one or several of these rankings while designing an interactive visualization. While we do not claim that making design decisions based on completion time and accuracy metrics is wrong, we emphasize that looking at metrics computed based on user behavior during the interaction cycle *(e.g., TRE, MDC)* can be helpful as well. Comparing interactive graphical encodings based on several metrics might help designers have a more holistic view of how well embedded interactions might work with certain encodings.

Incorporating the Interactive Graphical Encodings

If the decision is made to adapt the interactive graphical encodings in a visualization system, we suggest the following guidelines.

Making encodings interactive requires careful design considerations. Not every encoding used in a given visualization needs to be interactive. In cases where the chosen visual representation requires the use of an encoding with low performance, perhaps the use of traditional control panels for interaction is the better design decision. For example, visual representations that use *shading* or area as the primary method to encode data may be augmented with control panels to control the filtering or querying rather than embedded interaction *(e.g., geospatial choropleth maps)*. Instead, visual representations that use

effective encodings lend themselves better to incorporating directly on the encoding.

Provide additional feedback if accuracy is important. Providing additional feedback might be helpful to improve the performance of specific encodings. For example, during embedded interaction with *shading*, interaction performance might be improved by also showing exact values via textual overlay. Additionally, we could highlight the aspects of the encodings that contribute to the value change. For example, for *angular* encodings, we could highlight the angle subtended or the height between the two arcs. Similarly, for *area* encodings, we could highlight the width and height of the square to show the squared value. While we did not test the effectiveness of such potential design improvements in our study, these considerations could be of interest for future design and evaluation efforts.

Applications of Our Findings

In information visualization and visual analytics, the results of this study can be applied to inform the design of interactive legends [86, 70]. Interactive legends are controls that allow users to select or filter data by directly interacting with the graphical encodings used on the legends [86]. With the knowledge gained from this study, we suggest using the graphical encodings that have high accuracy *(e.g., length)* while designing interactive legends. Alternatively, legends using encodings with lower accuracy can provide additional feedback to users (e.g. textual values) to improve the accuracy of interaction. Another approach could be to resort to more conventional user interface widgets to perform tasks like filtering.

Another set of applications that could leverage the results of our study are graphical editing tools (e.g., Adobe Photoshop and Illustrator) and visualization authoring tools (e.g., Data-driven Guides [32]). Our findings can assist design decisions about where interactions must be enabled on the graphical encodings versus where additional widgets may be required. For example, to allow users to create a rectangle with a specific texture, these tools could let users adjust the dimensions of the rectangle using embedded interaction and provide additional widgets on a separate control panel.

Interaction Combines Perception and Manipulation

Although the methodology used in this study is different from that by Cleveland and McGill [78] due to our use of interactive magnitude adjustment, our ranking of the interactive graphical encodings produced a similar ranking. At a high level, our ranking follows that of the prior studies, with the exception of our results indicating a significant difference between length and angle (in terms of accuracy). An explanation for this similarity may be that manipulation and perception are not mutually exclusive, and input from perception continually influences interaction. Thus, the performance of interaction with an encoding might be connected to the perception of the encoding itself. If an encoding supports sheer perception well, it would also support interactivity well.

One possible follow-up research direction includes quantifying the distribution of how much of an effect both perception and manipulation have while interacting with a graphical encoding. To do so, the study design would need to directly control for, and decouple, perception from interaction. For example, this might involve shielding the participants' line of sight for the encoding they are asked to manipulate. However, this seems to be at odds with the design guidelines of embedded interaction, where users directly interact with handles superimposed on the graphical encodings Thus, performing a study where perception is intentionally excluded may limit the applicability of the results to informing the design of embedded interaction for visualization. However, the results of such a controlled study would reveal knowledge about the perception has on interaction.

Indirection, Compatibility, and Integration

The graphical encodings used in our study have different degrees of compatibility, indirection, and integration [87]. *Position*, *length*, *angle* and *distance* have low degrees of integration and indirection, and high degrees of compatibility. Thus, these encodings are more efficient than others encodings that have higher degrees of indirection and integration, and lower degrees of compatibility.

The differences in degrees of compatibility, indirection, and integration among various encodings may affect their performance. In particular, having a higher degree of indirection and lower degree of compatibility might decrease the performance of an encoding. One interesting avenue for continued research could be the investigation of effects of the parameters of this Instrumental Interaction framework proposed by Beaudouin-Lafon [87] on the performance of the encodings.

4.2.11 Limitations

Our results should be interpreted in the context of the specified encodings, adjustment orientations, target values, and tasks. We wanted to first gain a basic understanding of the rankings for simple interactive graphical encodings to see if and how they are different from the graphical perception results from prior studies [78, 79].

Since our study was online, we did not have control over users' physical devices. This decision was intentional so that participants could use input devices that they were familiar and comfortable with, but it also allows the possibility of effects due to system differences. We did record participants' operating system types and input devices, and we tested for effects using t tests. The results did not indicate a statistically significant effects due to mouse and trackpad for either accuracy ($t_{(33)} = 0.08$, $p = 0.93$, $power = 0.72$) or interaction time ($t_{(33)} = 0.49$, $p = 0.06$, $power = 0.38$). The near-significant trend in time due to interaction device reinforces the need to study the effect of interaction device in future studies.

We also did not find a significant effect of operating system for either performance time ($F_{(2,32)} = 3.02$, $p = 0.07$, $power = 0.95$) or accuracy ($F_{(2,32)} = 0.69$, $p = 0.51$, $power = 0.93$). Unfortunately, our collected logs did not contain information about participants' browser types and screen sizes; we suggest that future interaction-related online experiments take these two factors into account.

CHAPTER 5
TESTING FEASIBILITY OF VISUALIZATION BY DEMONSTRATION

RQ4: *How can we apply visualization by demonstration to design tools for data exploration?*

RQ5: *How effective is visualization by demonstration compare to traditional manual view specification?*

RQ6: *How to offer the benefits of both visualization by demonstration and manual view specification in a unified visualization tool?*

5.1 How can we apply visualization by demonstration to design tools for data exploration?

To show the feasibility of the visualization by demonstration concept, we implemented VisExemplar [6]. VisExemplar is a mixed-initiative data exploration prototype that allows users to explore their data using visualization by demonstration. VisExemplar uses a recommendation engine that generates potential changes and suggest them in response to the given demonstrations.

VisExemplar allows users to provide visual demonstrations by directly adjusting the graphical encodings used in visual representations *(e.g., users stacking data points in the shape of bars to convey their interest in a bar chart or placing two data points in desired positions along the x or y axis to demonstrate the attribute to map to the axis)*. Upon providing a demonstration, VisExemplar recommends a set of transformations (i.e. possible changes that can be made to the visualization) based on the given demonstration. We categorize these transformations into four main categories, described below.

- **Visualization Representation Transformations** change the current visualization

technique to a different visualization technique *(e.g., transforming from a scatterplot to a bar chart)*. To convey interest in transforming to a new visualization technique, users can manipulate the spatial encoding to create a spatial layout similar to the intended visualization. For example, users can stack two or more data points vertically or horizontally in a scatterplot to demonstrate their interest of switching to a vertical or horizontal bar chart.

- **Data Mapping Transformations** define mappings between graphical encodings and data attributes *(e.g., mapping color to an attribute)*. To convey interest in assigning a graphical encoding to a data attribute, users can manipulate the corresponding graphical encoding in the visual representations. For example, users could color one or more data points red to convey their interest in mapping color to a data attribute.

- **Axes Transformations** assign data attributes to axes of a visualization technique *(e.g., assigning an attribute to the x axis of a scatterplot)*. To assign new data attributes to axes, users can manipulate the corresponding graphical encoding or spatial encoding in the visual representations. For example, in a scatterplot, users could move one or more data points to a positions along an axis to demonstrate their interest in having a scatterplot in which the manipulated data point is close to the current coordinates, thus changing the attribute assigned to the axis. Alternatively, in a bar chart, users could change the length of one of more bars to demonstrate mapping a new attribute to the axis.

- **View Specification Transformations** change the view specifications without changing the underlying technique *(e.g., aggregation, average, sorting)*. For example, users could convey their interest in sorting a bar chart by dragging the longest bar in the current bar chart to the most left or right side of the axis.

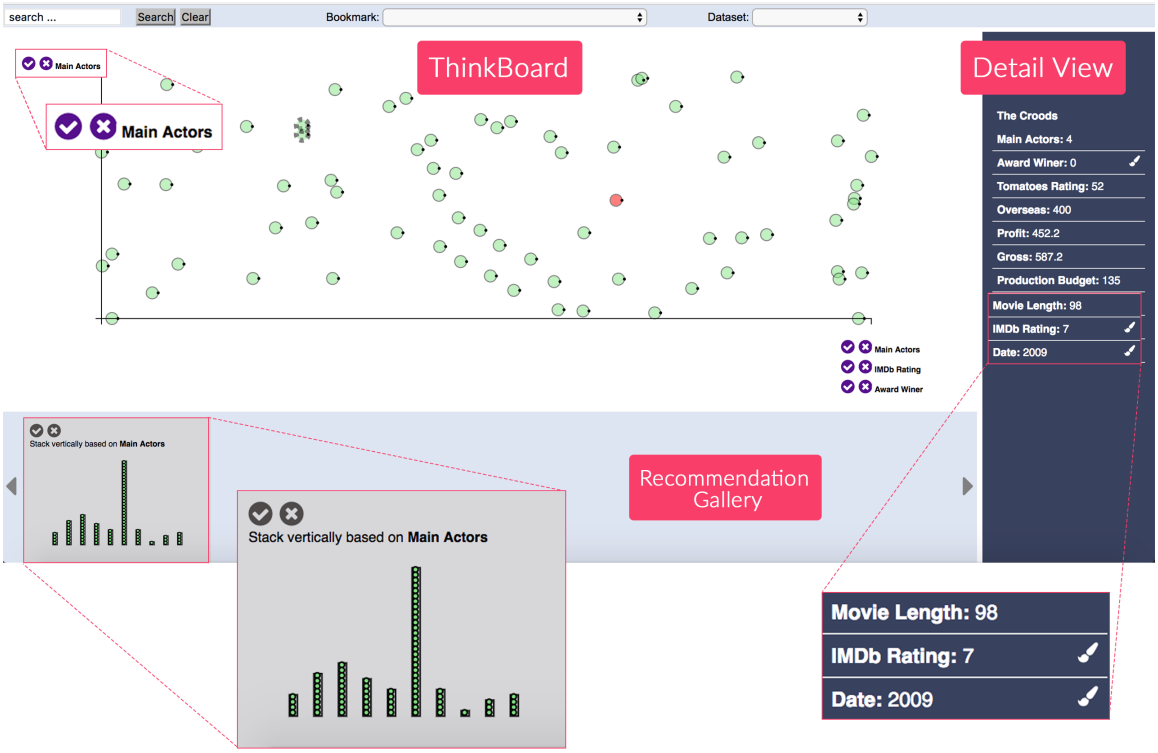


Figure 5.1: The VisExemplar user interface consists of a ThinkBoard, Recommendation Gallery, and a Detail View panel. ThinkBoard shows each data point as a circle. The Recommendation Gallery shows visualization technique transformations. The Detail View shows data details, and also recommended data mapping transformations.

5.1.1 The VisExemplar Interface

Figure 5.12 shows VisExemplar's interface, consist of a ThinkBoard, a Recommendation Gallery, and a Detail View panel. ThinkBoard is a thinking medium for users and allows them to construct their demonstrations through direct manipulation of the visual representation. Moreover, possible *Axes and View Specification transformations* be shown on the ThinkBoard. *Visual Representation transformations* will be presented in the Recommendation Gallery. The primary goal of the Detail View panel is to show details of selected data points. By hovering on a data point on the interaction board, the Detail View panel will show detail information related to that data point. See Figure 5.12 for more details. In addition, *Data Mapping transformations* will be shown as small icons on this panel.

VisExemplar provides an environment similar to a spatial workspace in which users provide demonstrations by manipulating the spatial and graphical encodings used in visual representations. In addition, to balance human and computer effort during data exploration process, VisExemplar suggests variety of possible relevant transformations in the form of *Visual Representations transformations*, *Data Mapping transformations*, *Axes transformations*, and *View Specification transformations*. Depending on the transformation type, they will be shown in different forms and locations on the interface. VisExemplar allows users to provide demonstrations by directly manipulating the data points in a visual representation. Finally VisExemplar uses different methods for showing recommendations to the user. First, visual representation transformations are shown as thumbnails below the ThinkBoard. These are ordered and colored based on their computed relevance to the visual demonstration. Data mapping transformations are shown as icons in the detail panel, and axes transformations on the axes. This helps users browse the possible space of transformations and interpret their result.

5.1.2 Transformations Supported in VisExemplar

VisExemplar currently supports four categories of transformations/changes.

Visual Representation Transformation. VisExemplar currently supports transformations from bar charts to a scatterplots and vice versa. Recommended *Visual Representation Transformations* will be shown on the Recommendation Gallery. Each recommended transformation is shown as a thumbnail in the gallery. Users can explore different transformations by scrolling through gallery. Each thumbnail consists of a textual explanation describing what the visualization in the thumbnail is showing *(e.g., Stack vertically based on Cylinder)* and a visualization which gives an overview of the transformation. We decided to show this type of recommendation as thumbnails because during the design process we found it difficult to imagine the resulting changes from one visual representation to another without seeing the resulting view. Relevance of this type of transformations is dually encoded by color and position. By default, we use a light gray color as background for recommended transformations in the gallery. We show the relevance of the recommended transformations by adjusting the darkness of the background color, the lighter the background color the lower the relevance. In addition, the recommended transformations are ordered left to right based on relevant (left being highest). Figures 5.2 indicates an example of *Visual Representation transformations* in VisExemplar.

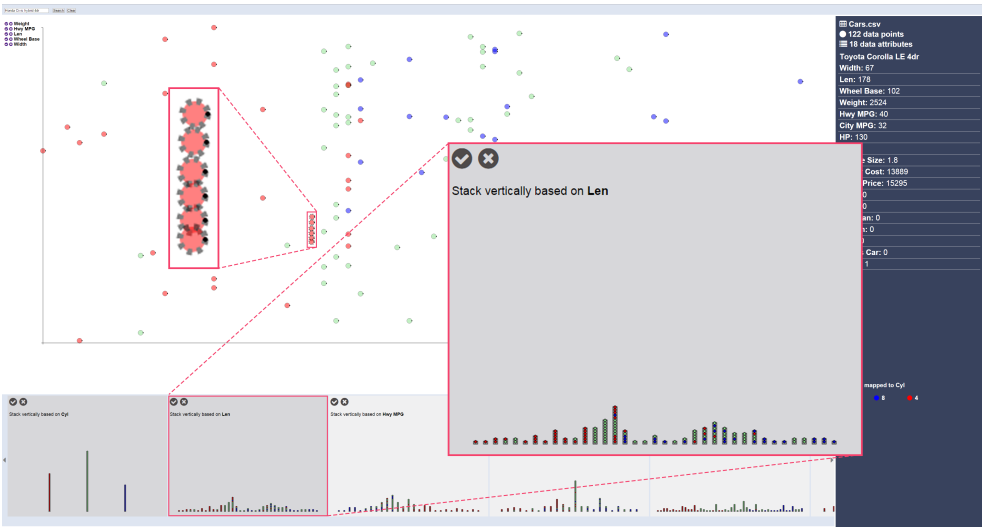


Figure 5.2: An example of Visual representation Transformation. Selected a bar chart where x axis assigned to Len and the y axis as the number of cars.

Data Mapping Transformation: The current version of VisExemplar supports color and size encodings. These types of transformations are shown as small icons on Detail View panel corresponding to the attribute which is being recommended to map to the visual demonstration. We decided to show this type of transformation on the Detail View panel since each icon is located beside corresponding data attributes. For those recommended data attributes that can be assigned to color, a small brushing icon (🖌) will appear near the data attributes on the detail panel. Similarly, the system recommends data attributes to be mapped to size by showing a small expand icon (⤢) beside the appropriate data attributes on the Detail View panel. The background color of the data attributes on hover shows the relevance. The lighter the background color the lower the relevance. Hovering on the recommended data attributes will also show a preview of resulting changes. A user can apply any of the recommended transformations by double clicking on the suggested data attribute. Figure 5.3-(d) shows examples of *Data Mapping transformations* in VisExemplar.

Axes Transformation. We show *Axes transformations* directly on the corresponding

axes in the ThinkBoard. In the early stages of our design process, we noticed that it is easier to understand the meaning of these type of transformations when they are located close to the corresponding axis. For this type of transformation, the position of the data attributes beside each axis show their relevance. The higher the data attribute the higher the relevance. Figure 5.4-(b) shows examples of *Axes transformations* in VisExemplar.

View Specification Transformation. This type of transformation is shown on the ThinkBoard below the visualization technique. One of these transformations that VisExemplar currently supports is sorting the bar chart in ascending or descending order (See Figure 5.5).

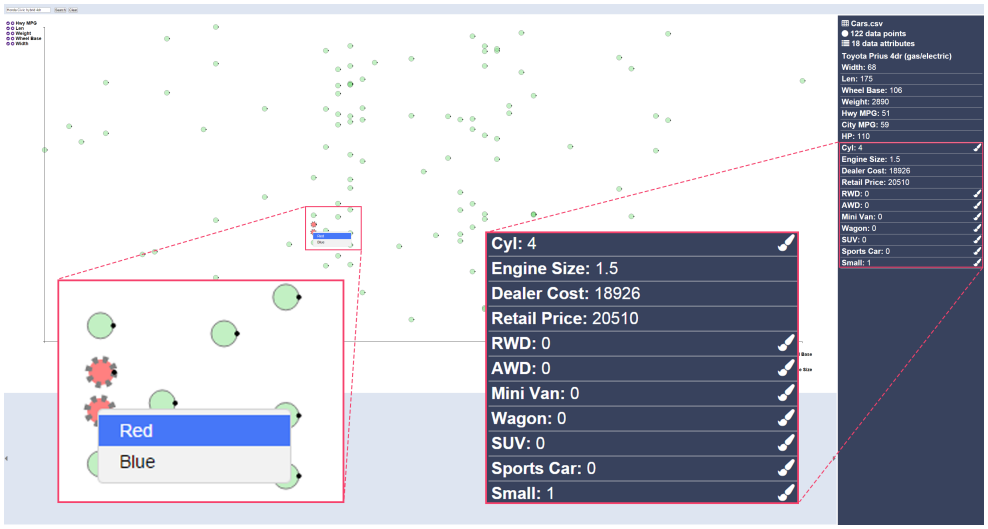


Figure 5.3: An example of Data Mapping Transformation. Colored two cars. The system recommended data attributes that can be mapped to color.

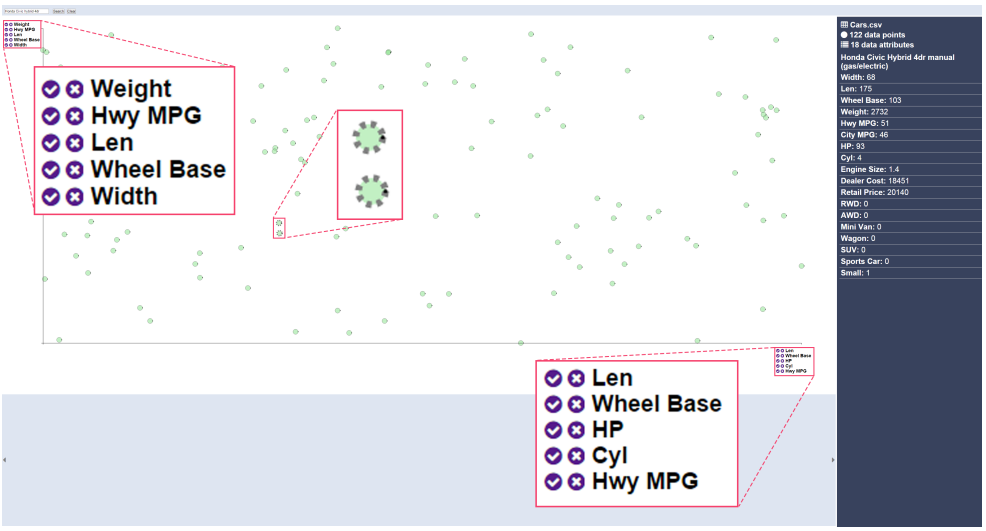


Figure 5.4: An example of Axes Transformation. Dragged two cars close to each other. The system recommended options for assigning to the x and y axis.

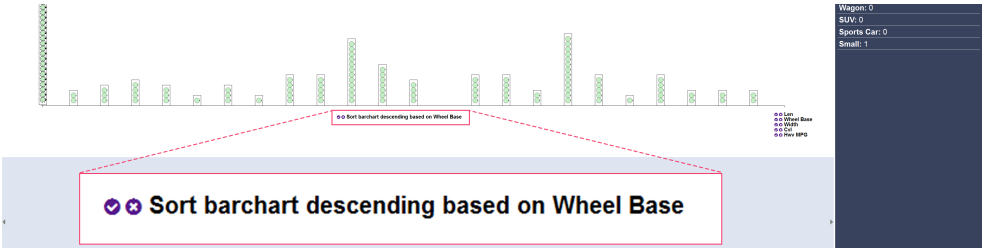


Figure 5.5: An example of a view specification transformation.

5.1.3 Recommendation Engine

Notice: At the time of developing VisExemplar, we abstracted intent and transformation functions and called it *"intent functions"*. Since then we thought about the underlying process in visualization by demonstration systems more deeply and decided to decouple intent and transformation functions. As such, intent function in this subsection refers to the combination of intent and transformation functions described earlier. When a user performs an interaction with the visual representation and generates a visual demonstration, the recommendation engine of our system accepts the interaction as input, and produces the recommendations as outpout for transformations mentioned in Section 5.1.2. VisEx-

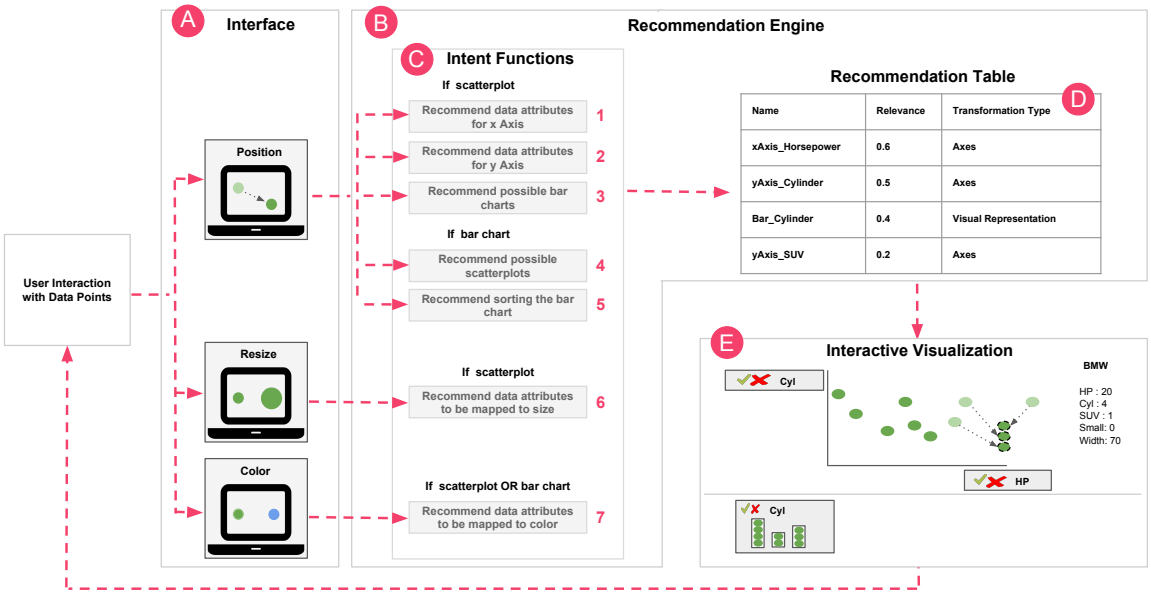


Figure 5.6: VisExemplar's Low-level Architecture. A) Recommendation engine takes user interactions as input. B) A series of intent functions drive the recommendation table. C) Direct manipulation of each encoding will invoke a series of intent functions related to that specific encoding. D) Recommendation Table will be updated after each interaction and stores a ranked list of potential transformations. E) The updated recommendation table feeds the recommendations in the user interface.

emplar allows direct manipulation of three encodings of data points including position, color, and size (see Figure 5.6-(A)). Direct manipulation of each encoding will invoke a series of intent functions related to that specific encoding (see Figure 5.6-(C)). Based on the demonstrations provided, the intent functions determine which transformations are most relevant. VisExemplar contains seven intent functions. All related intent functions are checked against every interaction. For example, by directly re-positioning data points in a scatterplot to new x coordinates, one of the intent functions which will be invoked is the *assigning X axis* function. Considering the points that have been moved, the system then recommends potential data attributes for the x axis that would result in a scatterplot where the moved data points would be as close as possible to the new x coordinates (see Figure 5.6).

As a result of each interaction, the recommendation engine will update the recommendation table (see Figure 5.6-(D)). The recommendation table consists of a set of potential

transformations. Each row of the table represents a potential transformation. Each transformation consists of a name *(e.g., xAxis_Cylinder)*, relevance, and location on the interface. The table will be created only once and will be updated after each interaction. The relevance value for each transformation indicates the number of times the transformation is generated. The relevance value is normalized to a range of $[0, 1]$ and the table is updated to contain the normalized relevance value for each transformation. The system only shows transformations with relevance above 0.3. Upon accepting a transformation the changes are applied and relevance values of all recommendations reset. The transformation type column dictates where each transformation is shown in the interface.

The recommendation engine then passes the recommendation table to the interface. The interface will update the visualization based on the given recommendation table; See Figure 5.6-(E).

Intent Functions

Depending on the interaction, any of seven currently-supported intent functions might be invoked (see Figure 5.6-(C)). For example, changing the position of a data point could invoke functions 1, 2, 3 or functions 4 or 5, depending on current state of the visualization (scatterplot or bar chart). If the current visualization is a scatterplot, then resizing a data point invokes intent function 6. Recoloring a data point will invoke intent function 7 regardless of the current state of visualization. Below we explain how each of these functions work. Full support of Visualization by Demonstration will require additional intent functions and our system design supports this extensibility.

The notations used in this section are summarized in Table 5.1. We refer to the data generally in normalized form, i.e. scaled into the interval $[0, 1]$ by attribute:

$$\tilde{d}_{ij} = \frac{d_{ij} - \min(\vec{d}_{\cdot j})}{\max(\vec{d}_{\cdot j}) - \min(\vec{d}_{\cdot j})}, \tag{5.1}$$

where $\min(\vec{v})$ and $\max(\vec{v})$ indicate the smallest and largest element respectively of vector

$\vec{v}$.

Position

Depending on the type of the current visualization and user interactions, upon changing the positions of data points, intent functions 1, 2, or 3 (for a scatterplot), or functions 4 or 5 (for a bar chart) are triggered. In this section, we describe how each of these five functions will be triggered after moving the data points.

If the current visualization is a scatterplot, upon the movement of the point, the system either recommends changing the Axes (changing the attributes assigned to x or y-axis) or changing the visual representation to a bar chart.

Intent Function 1 (Figure 5.6-(C)-1): After a position-changing interaction, the system searches for data attributes to assign to the axes in a scatterplot based on the positions of the moved data points. For example, in Figure 5.7, the user starts with a scatterplot whose x-axis represents miles-per-gallon (MPG). When the user moves a data point (black arrow in Figure 5.7 (a)), the x coordinates of the scatterplot no longer map to MPG. Rather, their position is better aligned with the length attribute (Figure 5.7 (b)). In this case, the system recommends assigning length to the x-axis.

In detail, we linearly normalize coordinate vectors $\vec{x}$ and $\vec{y}$ into $\tilde{x}$ and $\tilde{y}$ in the same way as Eq. 5.1, so that they are in the range of $[0, 1]$. As a result, coordinate values and data attributes values are in the same scale. Then, we find an data attribute that minimizes the sum of squared differences between normalized coordinate values and data attribute values of data points. In other words, the system recommends attribute j^* to be assigned to x-axis such that

$$j^* = argmin_j \left\| \tilde{\vec{d}}_{\cdot j} - \tilde{\vec{x}} \right\|_2^2.$$

Intent Function 2 (Figure 5.6-(C)-2): In the same way, the system also recommends data attribute(s) for the y-axis of a scatterplot.

Intent Function 3 (Figure 5.6-(C)-3): If a position-changing interaction results in more

Table 5.1: Notation used in this chapter.

NOTATION	DESCRIPTION
n	The number of data points
m	The number of data attributes
$\vec{d_i}$	The i-th data point
d_{ij}	The j-th attribute value of the i-th data point
$\vec{d_{\cdot j}}$	The j-th attribute vector
$\vec{x}$, $\vec{y}$	The vector of plotted coordinates in the x (or y) axis
x_i, y_i	The plotted value of the i-th data point in $\vec{x}$ or $\vec{y}$
c_i	The plotted color of the i-th data point
a_i	The plotted area of the i-th data point
$\tilde{v}$	The normalized value of vector v into the interval $[0, 1]$
$\|S\|$	The number of elements in a set S

than three data points lined up in a row, the system then checks if the three data points are within a specified distance from each other. If so, the system detects user interest for transforming the visualization into a bar chart. In other words, the system interprets the visual demonstration of "stacked data points" as a user's interest in transition to a bar chart. The function then computes common data attributes shared by the aligned data points. Finally, it updates the recommendation table. For instance, if a user places four data points that represent SUVs with six cylinders on top of each other, the system recommends a bar chart by car type *(e.g., SUVs, sports cars, etc)* and a bar chart by the number of cylinders.

If the current visualization is a bar chart, users can move the data points inside each bar or move a bar itself within the bar chart. Note that each bar is shown as a visible box which contain a set of corresponding data points.

Intent Function 4 (Figure 5.6-(C)-4): If the user drags a data point out of a bar in a bar chart, the system interprets it as a demonstration of changing the visual representation to a scatterplot. After the user drags out two or more data points, the system searches for data

attributes that will be assigned to axes of a new scatterplot (using a similar method to the Intent Functions 1 and 2). Based on the current x and y coordinates of the moved points, the system recommends potential x and y axes so that the new scatterplot representations of the moved points would be similar to the current user-defined positions. The recommendations with previews will be shown in the Recommendation Gallery.

Intent Function 5 (Figure 5.6-(C)-5): Users can drag and drop any of the bars shown in a bar chart. If the user drags the longest bar in the bar chart to the left most side of the bar chart the system recommends sorting the bar chart descending. If the user drags the longest bar in the bar chart to the right most side of the bar chart the system recommends sorting the bar chart in the ascending order.

Resizing

Users can resize data points any time during the data exploration process. Users can adjust the size by dragging a small handle (tiny black circle) on the perimeter of the data point.

Intent Function 6 (Figure 5.6-(C)-6): When a user resizes a data point in a scatterplot, the system interprets it as the user's interest in encoding a data attribute to the demonstrated sizes of data points. In order to provide enough information to the system for recommending a mapping from a data attribute to data point sizes, the user has to resize two or more data points. The system shows recommended transformations that are above a developer-defined threshold by showing a expand icon (↙↗) beside those attributes on the Detail View. Specifically, we first normalize the sizes of data points the user has adjusted in a way that reflects the fact that the minimum is nonzero and that we are seeking changes in the same direction (bigger or smaller) by setting the default drawing value to 0.5. We calculate a scaled value $\tilde{a_i}$ $\forall i \in S$ (the set of modified points) as follows:

$$\tilde{a_i} = \begin{cases} \frac{1}{2}\left(1 + \frac{a_i - a_0}{\max(a) - a_0}\right) & \text{if } a_i > a_0 \\ \frac{1}{2}\left(\frac{a_i - \min(a)}{a_0 - \min(a)}\right) & \text{if } a_i < a_0 \end{cases},$$

where a_i is the current plotted size of the i-th data point, a_0 is the default plotting size, and the max and min functions return the predetermined maximum and minimum drawing sizes for points in the visualization. The system recommends attribute j^* for size encoding such that

$$j^* = argmin_j \sum_{i \in S} (\tilde{d}_{ij} - \tilde{a}_i)^2,$$

where S is the set of resized data points.

Recoloring

Users can recolor a data point by right clicking on it and picking a color from the pop-up menu. VisExemplar currently supports three colors: red, blue, and green (default).

Intent Function 7 (Figure 5.6-(C)-7): We now describe the intent function triggered by changing the colors of data points. For coloring interactions, we consider categorical attributes only and ignore numerical attributes. We define an attribute as categorical if the number of unique values present is fewer than ten and also fewer than half of the number of data points. That is, attribute j is categorical if and only if

$$|\text{unique}(\vec{d}_{\cdot j})| \leq \min(10, \frac{n}{2})$$

If the user changes the color of one data point, the system makes a recommendation for each categorical attribute, suggesting applying the same recoloring to all other points sharing the value. For example, if the user changes the color of an AWD sedan with 6 cylinders to red, the system recommends three options: coloring all AWD vehicles red, coloring all sedans red, or coloring all 6-cylinder cars red.

When a user colors two or more data points, two conditions are checked to find the appropriate mapping. The first checks for positive correspondence between a data attribute and the assigned color. The condition is satisfied for an attribute if all points given the

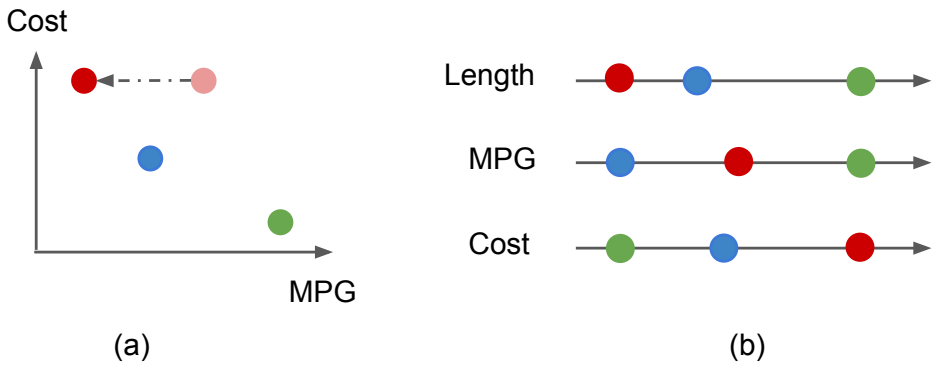


Figure 5.7: Position-changing interaction. (a) A scatterplot with MPG as x-axis and cost as y-axis. A user moves a red data point. As a result, the system recommends assigning length attribute to x-axis. (b) A visual representation of data distributions for potential data attributes.

same color also have the same value for that attribute. The second condition tests that whenever two points have different colors, they have different attribute values. Given a set of indices of k modified points $i_1, \ldots, i_k$, the two conditions on attribute j are, for all pairs $(k, p) : k, p \in i_1, \ldots, i_k$ and $k \neq p$:

$$\text{Condition I: If } c_k = c_p, \text{ then } d_{kj} = d_{pj}$$

$$\text{Condition II: If } c_k \neq c_p, \text{ then } d_{kj} \neq d_{pj}$$

If all the user-colored data points have the same color, the system checks every categorical data attribute to see if its attribute values of the colored data points are the same using Condition I. The system then recommends all data attributes that meet Condition I. For example, suppose the user colors an AWD sedan with 6 cylinders and a non-AWD sedan with 6 cylinders red. Since both cars are 6-cylinder sedans, two attributes, car-body-type (which includes sedan) and number-of-cylinders, satisfy Condition I. The system recommends two options: coloring all sedans red or coloring all 6-cylinder cars red.

If the data points are colored with two or more colors, the system uses both conditions to evaluate attribute mappings. The two or more colors specify not only the mapping of color to an attribute, but the assignment of values of that attribute to one specific color. First, Condition I is applied across each subset of the re-colored points that have been assigned

the same color. This discovers the data attributes shared by each colored group of the modified points. Second, Condition II is applied with the attributes revealed by Condition I to find which attributes can account for the differences across color groups. For example, suppose the user re-colors three data points: two representing cars with attribute values given by the tuples (AWD, sedan, 6 cylinders) and (FWD, sedan, 6 cylinders) are colored red; the third, (AWD, sedan, 4 cylinders), is colored blue. In the red group, both body-type *(e.g., sedan)* and number-of-cylinders satisfy Condition I. However, when Condition II is checked, we see that only the cylinder attribute satisfies both conditions. The system recommends mapping the cylinders attribute to color by coloring 6-cylinder cars red and 4-cylinder cars blue. A brush icon (🖌) beside each of the candidate attributes on the Detail View shows the recommendation to the user.

5.2 How effective is visualization by demonstration compare to traditional manual view specification?

As we discussed in section 3, a commonly used interaction paradigm in most visualization tools is *manual view specification (MVS)*. Tools implementing MVS often require users to manually specify the desired mappings through GUI operations on collections of visual properties and data attributes that are presented visually on control panels. In this paradigm, users are responsible for specification of the mappings, while the system computes the resulting view. MVS frequently used in successful visualization tools such as Tableau [27] and many more.

Visualization by demonstration advocates for a different process of visualization construction. Instead of specifying mappings between data attributes and visual representations directly, visualization by demonstration lets users demonstrate partial mappings or changes to the output (the visualization). From these given demonstrations, the system interprets user intentions, and recommends potential mappings.

Although both MVS and VbD offer iterative processes for creating visualizations, they

have fundamental differences:

- MVS uses a different process compared to VbD. MVS requires users to specify visualization techniques, mappings, and parameters. In contrast, VbD requires users to provide visual demonstrations of incremental changes to the visualization. It then recommends potential visualization techniques, mappings and parameters from the given demonstrations. Thus, while the result of both is a visualization and the corresponding specifications, users follow a different process to get there (see Figure **??**).

- MVS introduces interface elements such as menus and dialog boxes that act as mediators between users and the visual representation. In this case, the interface is an intermediary between the users and the visual representation. In contrast, in VbD the interface is itself a visual representation. The user can act on the visual representation rather than external interface elements. While the majority of user interaction is on the visual representation, VbD still makes use of some interface elements for accepting or rejecting the recommendations. In this case, there are fewer intermediary elements between the user and visual representation: the user can demonstrate intentions by directly manipulating the encodings used in a visual representation.

Many visualization tools have implemented the MVS paradigm. These tools have been successful in easing the processes of visualization construction and data exploration [88]. They allow users to interactively change parameters to construct visualizations and explore data instead of using programming. However, when new interaction paradigms such as visualization by demonstration are created, it raises a number of intriguing questions including: *How do interaction paradigms enable different visualization construction processes? How effective are each of these interaction paradigms for specific tasks? Which interaction paradigms do people prefer when constructing visualizations and exploring data?* Understanding the differences and trade-offs between various interaction paradigms

and how they are used for specific tasks can help designers and developers make informed decisions about adapting these paradigms in visualization tools.

To investigate trade-offs between the two interaction paradigms, so far we have conducted an exploratory study to investigate how these interaction paradigms affect strategies that people follow while exploring their data, the common patterns that appear, design choices, and the challenges people encountered using each tool [13].

5.2.1 Differences Between MVS and VbD

Although both MVS and VbD offer iterative processes for creating visualizations, they have fundamental differences [13]. One way to look at these differences is to consider the dimensions of **visualization construction model** and **number of intermediary interface elements** [13]. In terms of the **visualization construction model**, MVS requires people to specify visualization parameters (e.g., mappings). In contrast, VbD requires people to provide visual demonstrations. It then recommends potential visualization techniques, mappings and parameters based on the system's interpretation of the demonstrations. In terms of **number of intermediary interface elements**, MVS introduces interface elements (or instruments [76]) such as menus and dialog boxes that act as mediators between the user and the visual representation. In contrast, VbD lets people interact directly with the visual representation rather than external interface elements, as much as possible. Although implementations of VbD usually rely on some external interface elements, for example for accepting or rejecting the recommendations, the number of required interface elements is smaller than with MVS.

MVS and VbD are also different when considering the dimensions of **agency** and **granularity** [89]. **Agency** refers to *who* is responsible for carrying out the visualization construction process: the user or the tool. With most MVS tools such as Polestar, Many Eyes, and Spotfire, the agency tends to be more on the user side than on the tool side [89] because design decisions are mostly driven by the user. With VbD, agency is shared between the

user and the tool because of the automation that occurs as part of recommending visualizations based on user demonstrations. **Granularity** refers to the level at which the tool enables the manipulation of both data and visual representations. Most MVS tools have a *coarse* granularity [89], as they let users operate on data attributes and manipulate groups of marks (e.g., LARK [90]). Tools that have a *fine* granularity like iVolver [91] let users access individual data values and manipulate individual marks. VbD has a finer level of granularity than MVS, as it lets users directly manipulate individual graphical encodings rather than attributes.

5.2.2 Study Design and Choice of Visualization Tools

In our study, we used two visualization tools, VisExemplar [6] and Polestar [36], which satisfied two requirements. One is that each had to clearly embody one of the two interaction paradigms (either MVS or VbD). The other is that to have a fair comparison between these two paradigms, users had to be able to learn and use the system within the duration of the experiment. As each tool adopted only one of the two interaction paradigms, we compared VisExemplar, which incorporates visualization by demonstration and Polestar, which incorporates MVS. Although a variety of commercial tools incorporate the MVS paradigm, we decided to use the less complicated visualization tool, Polestar, to control for external factors that might affect the study. The functionality of Polestar is tightly scoped and intentionaly limited to control for these potential confounds. Also, Polestar has previously been used as a control condition in previous studies [34, 92].

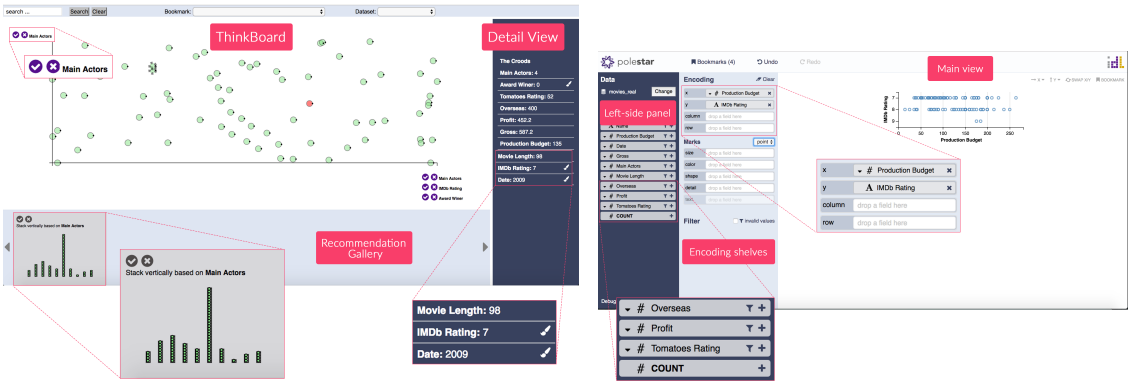


Figure 5.8: Left: A screenshot of VisExemplar, which implements Visualization by Demonstration. **Right:** A screenshot of Polestar, which implements MVS.

5.2.3 Pilot Study

Our study design was shaped by a pilot study. Our goal in the pilot study was to capture potential flaws in our main study design and understand if the datasets were appropriate for a 20 minutes of data exploration session during our main experiment. For the pilot study, we recruited six participants (4 male, 2 female) between 23 and 28 years old. Participants had backgrounds in computer science and social science. All of the participants had experience creating visualizations using Microsoft Excel. One of the participants also had experience in creating visualizations using Tableau.

We randomly assigned three of the participants to work with VisExemplar and the others to work with Polestar. We first introduced each tool to the participants and trained them for 10 minutes. We then asked participants to perform eight trial tasks using the tool. The tasks were to construct and refine visualizations (e.g., sort the given bar chart in ascending order, switch between visualization techniques, assign a data attribute to the x-axis, etc.). Tasks for trial sessions were designed using dataset containing cars and various attributes describing them [67].

Once participants completed the tasks, we asked them to work with the visualizations tool for 20 minutes to explore a dataset about cameras [68]. The participants were asked to

verbalize their exploration process. We recorded 170 minutes of participants' visualization processes in the form of video screen captures. After recording the video screen captures. The analysis of the screen recordings gave us an initial understanding of the common types of specifications participants created using each paradigm and the difficulties they encountered while exploring their data using each paradigm.

We found that Polestar worked better for cases where participants knew the exact information required for constructing or refining a visualization. In most cases, this came down to their task being defined in terms of data attributes. For example, for cases where participants knew exactly which data attribute should be mapped to which visual encoding or axis, they could do so through direct specification via the control panel. In contrast, VisExemplar worked better for cases where participants had some ideas of how the final visualization should look like, had some knowledge of the data items, but were less familiar with the data attributes or visualization terminology used in the control panel required for completing, constructing, or refining the visualization. For example, cases where the participants knew that they want a scatterplot where more expensive cameras are bigger, or even where they found specific cameras that they wanted a certain color without a formal understanding of what attributes create that mapping.

We initially decided to use the Cameras [68] datasets in our main study. However, based on the pilot study, we found that participants were not familiar with many data attributes used in the Cameras dataset (or cameras in general), so they tended not to explore for as long. Participants tended not to examine attributes of the data they were not familiar with and therefore made more impetuous decisions. We instead decided to use a Movies dataset [68] that provides details for 335 movies released from 2007 to 2012, and contains 12 data attributes. We selected the Movies dataset for our main experiment based on two considerations. First, the dataset contained enough data attributes to support 20 minutes data exploration. Second, the participants were unfamiliar with the content of the dataset but familiar with the meaning of the data attributes used in the dataset.

Perhaps more importantly than its results, our pilot study helped us to focus our research on a key question: How do these two interaction paradigms affect the visualization construction and visual data exploration process? To explore this question more in-depth, we conducted the following in-depth study.

5.2.4 Study Design

We conducted a two-phase study. In the first phase, we studied how well each interaction paradigm supports visualization construction in a more controlled setting. We measured the effectiveness (performance time and accuracy) of each paradigm for 16 tasks. In the second phase, we investigated how well visualization construction is supported by each paradigm in a more realistic scenario. We conducted a think-aloud exploratory observational study in a laboratory setting where participants were asked to use a visualization tool to explore a dataset. Study materials are available in the supplemental materials.

Participants and Setting

We recruited 16 participants (9 female and 7 male), between 21 and 32 years old. The participants were undergraduate and graduate science and engineering students. None of them had participated in the pilot. All participants reported to be familiar with reading and creating visualizations using existing tools such as MS Excel (16), D3.js (2), and Tableau (1). During the entire study, participants used a computer with a 13 inch screen. Two of the participants took the undergraduate level information visualization course taught in our university. The study took about 1 hour to complete and participants were compensated with a $10 Amazon gift card.

5.2.5 Phase 1: Controlled Experiment

In this phase, we examined the effectiveness of each interaction paradigm for creating visualizations in a controlled setting. We used a mixed design with the tool as a between-

subjects factor. 16 subjects participated in our study and were randomly assigned to one of the visualization tools (8 participants per visualization tool). Each participant worked with just one of the visualization tools.

Tasks

To select tasks for our study, we first interacted with both VisExemplar and Polestar exploring different ways in which they support visualization construction. This resulted in a list of 25 visualization construction tasks (e.g., assign a data attribute to size and color of data points). We also reviewed taxonomies of tasks commonly used for interactive visualization construction (e.g., [93, 3, 94, 95]). Considering our experiences with these tools and our knowledge from these taxonomies, we then assigned these tasks into one of four categories according to the type of changes they make to a visualization.

- **Mapping data attributes to the axes:** This category of task requires users to assign data attributes to either one or both axes of a visualization.
- **Mapping data attributes to mark properties:** This category of task requires users to map a data attribute to a mark property.
- **Switching between visualization techniques:** This type of task requires users to change from one visualization technique to a different visualization technique.
- **Reconfiguring a visualization:** This category of task requires users to change the view specification of a visualization without changing the underlying technique and mappings.

For this phase of our study, we designed 16 tasks for participants to perform (4 categories of tasks $\times$ 2 phrasing methods $\times$ 2 trials). Each participant performed all 16 tasks using one of the visualization tools. The phrasings of tasks (abstract and specific) are based on how participants verbalized their goals during the think aloud protocol of the pilot study.

Table 5.3 shows four category of tasks used in our study. For each task type, we included two equivalent task phrasings.

Hypotheses

Since the two interaction paradigms have their own characteristics, we expect that each paradigm will have its advantages and disadvantages. Based on our pilot studies and experience with both paradigms, we considered the following hypotheses for our study:

- **H1:** We hypothesize that using Polestar, participants map data attributes to axes and switch from one visualization technique to another significantly faster and more accurately than VisExemplar. However, mapping visualization encodings to data attributes and reconfiguring visualizations would be significantly faster and more accurate using VisExemplar.
- **H2:** We expect Polestar to have better task performance (faster and higher accuracy) for tasks phrased using the specific method and VisExemplar to have better performance for those phrased more abstractly.

Procedure

Training. Before starting the main experiment, participants were briefed about the purpose of the study. At this stage, the participants were also asked to answer some demographic questions (e.g., age, sex, and prior experience in creating visualizations). Each participant was asked to work with one of the visualization tools. We first walked the participants through the training session to familiarize them with the study. As our participants had no prior experience using these particular tools, we reduced their initial learning time by offering a brief introduction to the tool they would use. To prevent inconsistencies in the training session, we asked participants to watch a tutorial video of the visualization tool. The video walked the participants through different features and interactions provided by

the tool. The participants were allowed to watch the video as many times as they want. After watching the video, we asked participants to work with each tool for 10 minutes. In addition, we encouraged participants to ask as many questions as they want during this stage. We then asked participants to perform 8 training tasks (4 types of tasks $\times$ 2 phrasing methods $\times$ 1 trial). The participants were not allowed to move to the next training question unless they answered the question correctly.

Main Study. In this phase, each participant performed 16 visualization construction tasks: 4 types of tasks $\times$ 2 phrasing methods $\times$ 2 trials. All tasks were printed on a sheet of paper. Each time the interviewer selected a task randomly and asked the participants to perform the task as fast and accurately as possible. Before performing each task, participants were given a visualization as a starting point. This way we made sure that all the participants performed each task starting from the same visualization. We measured participants performance time and accuracy. To design tasks for this phase, we used the Cars [67] dataset. The Cars dataset [67] provides details for 407 new cars and trucks for the year 2004. This dataset contains 18 data attributes describing each car.

Data Analysis

To address our first hypothesis (**H1**), we tested how the different tasks were performed using each interaction paradigm in terms of time. We initially planned to take into account both performance time and accuracy in our analysis. However, the participants performed all the tasks correctly using both paradigms, so we excluded accuracy from our analysis. We first calculated separate mean performance time for all trials. For each participant, we averaged outcome values of trials for each type of task. We then conducted a mixed analysis of variance (ANOVA) to test for differences among the four types of tasks (within-subjects factor) using two interaction paradigms (between-subjects factor). The main effect of interaction paradigm indicates which paradigm produces the best performance, regardless of

the task. The task $\times$ paradigm interaction indicated whether a particular paradigm works better with a particular task.

To address our second hypothesis (**H2**), we conducted the second mixed ANOVA to test for differences among the two phrasing methods of constructing visualizations (within-subjects factor) using two interaction paradigms (between-subjects factor). In particular, we were interested in interaction between the two different phrasing methods for goals and interaction paradigms (phrasing methods $\times$ interaction paradigm). Investigating the interaction between phrasing methods and interaction paradigms indicated whether a particular paradigm works better with a specific method of phrasing goals (abstract and specific). Thus, we averaged outcome values of trials for each participant.

Before testing, we checked that the collected data met the assumptions of appropriate statistical tests. The assumption of normality was satisfied for parametric testing, but Mauchly's Test of Sphericity indicated that the assumption of sphericity had been violated for time. To address this issue, we report test results with corrected degrees of freedom using Greenhouse-Geisser estimates for $\epsilon < 0.75$ and otherwise with Huynh-Feldt correction.

Results

We found a significant effect of performance time for interaction paradigm ($F(1, 14) = 19.6, p < 0.05$) with a slightly large effect size ($\eta_p^2 = 0.63$). Overall, average task completion time across all tasks showed that Polestar was three seconds faster than VisExemplar. Tools such as Polestar that implement MVS are fast and accurate as they enable rapid and exact specification of the visual properties by incorporating a set of consistent user interface elements. We also found a significant interaction between paradigms and tasks for performance time ($F(1, 14) = 16.8, p < 0.05$) with a slightly large effect size ($\eta_p^2 = 0.56$). Our results show that participants mapped data attributes to axes significantly faster using Polestar compared to VisExemplar. We also found that Polestar was significantly faster

than VisExemplar in switching from one visualization to another. However, participants were significantly faster in mapping data attributes to encodings and reconfiguring visualizations using VisExemplar. Our results partially confirm our first hypothesis (**H1**). See Figure 5.9 for more details.

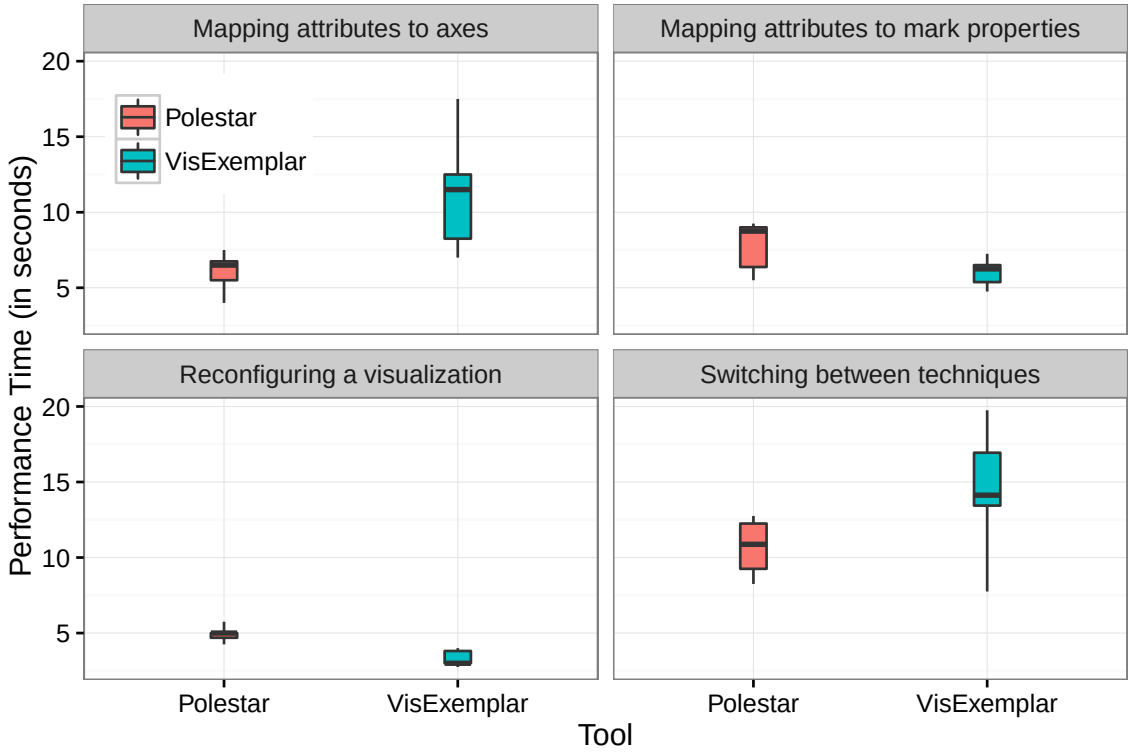


Figure 5.9: Average performance time of participants for each type of task using Polestar and VisExemplar.

We also found a significant interaction between paradigms and phrasing methods for performance time ($F(1, 14) = 34.5, p < 0.05$) with a large effect size ($\eta_p^2 = 0.71$). The participants performed tasks significantly faster using the abstract method than the specific method in VisExemplar ($p < 0.001$). Unlike VbD, the participants were significantly faster in performing tasks using the specific method than the abstract method in Polestar that implements MVS ($p < 0.05$). This confirms our second hypothesis (**H2**). See Figure 5.10.

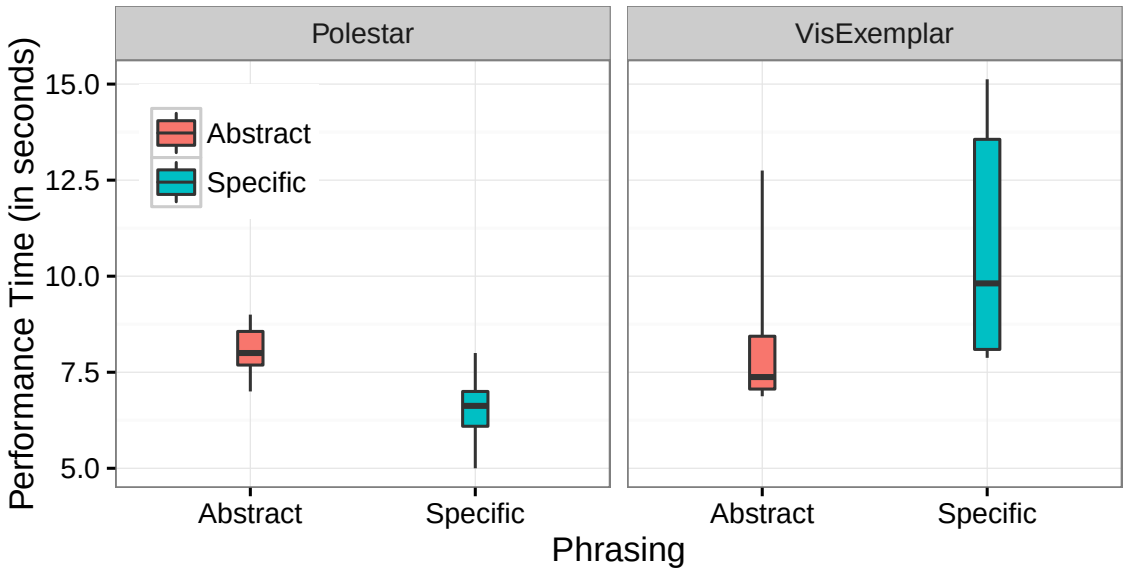


Figure 5.10: Average performance time of participants for each phrasing method using Polestar and VisExemplar.

5.2.6 Phase 2: Open-ended Exploration

In this phase, we conducted a think aloud exploratory observational study to understand how the participants use each interaction paradigm to construct visualizations in a more realistic scenario.

Procedure

Main Study. In this phase, the participants were asked to explore the Movies dataset [68] and look for interesting findings about the data. In particular, the participants were told to imagine their employer asked them to analyze the dataset using the visualization tool for 20 minutes and report their findings about the data. Participants were instructed to verbalize analytical questions they have about the data, the tasks they perform to answer those questions, and their answers to those questions in a think-aloud manner. In addition, we instructed them to come up with data-driven findings rather than making preconceived assumptions about the data. The participants were not allowed to ask any question during this phase. We tried to avoid interrupting the participants as much as possible during their

data exploration process. However, sometimes it was necessary to remind the participants that this is a think-aloud study and they need to verbalize their thoughts.

Follow-up Interview. We asked participants what they liked and disliked about the interaction paradigm. This was to allow the participants to convey their feedback and ideas and to solicit potentially unexpected insights.

Data Collection and Analysis

To analyze differences between the VbD and MVS conditions, we gathered several types of data. At the beginning of the training session, we used questionnaires to collect participant demographic and background information. During the main study, we took written notes of participants' interaction processes with the tools. We also screen- and audio-recorded the whole study.

To analyze the video and interview material, we followed guidelines provided by Creswell [74, p. 236] for analyzing qualitative data. We first transcribed data from the interviews. The coder (first author) then read the transcribed materials to obtain a general sense of the data and started thinking about organization. After reading the data, the first author identified the meaningful text segments and assigned a code word or phrase that accurately describes the meaning of the text segment. The coding process was an iterative process with three passes by a single coder in which the coder developed and refined the codes. During the coding phase, we mainly focused on processes of the participants in terms of **usage** (what types of visualization specifications were usually created using each interaction paradigm? What usage patterns exist for specific functionality?) and **barriers** (when and how difficulties happened while working with each paradigm?) For example, our codes included phrases such as "changing color", "changing size", and "stacking data points". Finally, we aggregated similar codes into themes, and assigned them labels. For example, we aggregated the "changing size" and "changing color" codes to create a "mapping a data attribute

to a visual property" theme. Finally, we identified frequently occurring codes and themes to form higher-level descriptions to discuss our findings.

Results

In this section, we categorize and discuss the findings of our study.

Interaction Behavior for Visualization Construction Tasks

We divided types of operations the participants performed during the entire visualization construction process into four categories of tasks discussed earlier (Section 5.1).

Mapping Data Attributes to Axes. Participants tended to map more data attributes to axes using Polestar (see Table 5.2). Four of the eight participants who worked with Polestar stated that the fast speed of the tool in mapping data attributes to axes have contributed to this advantage. On the other hand, in VisExemplar, five of the participants expressed difficulties in mapping data attributes to axes. To map a data attribute to an axis using VisExemplar, the participants had to position a few data points relative to their data attribute values. The system then recommended potential data attributes to be assigned to the axes. For example, one participant expressed how a large amount of effort was required for him to map a data attribute to an axis: *"You know it is hard to drag the points and track their values, [...] maybe you could somehow highlight the values [data attribute values] while moving the points to decrease users' cognitive load."*

Another challenge that three of the participants encountered while using VisExemplar was the accuracy of the data attributes suggested to be mapped to the axes. After providing visual demonstrations, the system searches for data attributes to recommend for mapping to the axes based on the user interaction. The recommendation engine prioritizes potential suggestions and shows those above a certain threshold. However, there might be cases where a user's expected data attribute is not among those recognized to be the most related

ones by the system. In such cases, users have to provide more demonstrations to help the system to interpret their intentions better. One of the participants mentioned her concern by saying: *"The recommendations on the axes don't always make sense to me. When I have an idea in mind like let me see how these [data attributes] compare, then when I don't see it in the options, I am kinda thrown off because at that point I am kinda doubting whether the way that I am thinking about it is wrong or whether I am doing something wrong with the system."*

Mapping Data Attributes to mark properties. The participants mapped more data attributes to mark properties using VisExemplar (see Table 5.2). To map a data attribute to size or color using VisExemplar, users could manipulate characteristics of a corresponding encoding in the visual representation. For example, users could color one or more data points red to convey their interest in mapping this specific color to a data attribute. The system then recommends a set of data attributes that can be mapped to color. During data analysis, we noted multiple interesting patterns.

In VisExemplar, participants found the process of recommending a subset of appropriate data attributes for mapping to color or size very interesting and helpful. For instance, one participant mentioned that: *"[...] coloring points was fast though. I can color one point and the system suggests a small set of attributes."* On the other hand, one of the participants who used Polestar stated: *"every time I need to skim through attributes on this panel [the panel showing data attributes], pick one, and drag it. It becomes hard to skim through all attributes if we have many of them [data attributes]."* Moreover, we saw an interesting pattern emerge when the participants did not intend to map any specific data attribute to an encoding but wanted to explore different mapping options by hovering on the recommended data attributes. For example regarding VisExemplar, one of the participants mentioned: *"... let's color one and look at recommendations [participant hovered on the recommended attributes to preview the results and explain their findings]."*

Table 5.2: Total and average number of times that participants performed each type of task using VisExemplar and Polestar.

TASK TYPE		VISEXEMPLAR	POLESTAR
Mapping attributes to axes	Total	53	108
	Avg	7.5	15.4
Mapping attributes to encodings	Total	55	26
	Avg	3.9	1.9
Switching between techniques	Total	12	25
	Avg	1.4	3.5
Reconfiguring a visualization	Total	7	9
	Avg	0.8	1

We also noted that the participants felt more control over the tool when they were mapping data attributes to mark properties using VisExemplar. One participant expressed his feeling of having control by saying: *"I like that I can color it here [coloring the glyph], I feel like I have control over the circles [data points]"*. This is potentially because VbD advocates for increasing the level of "interaction directness" [87] by enabling the users to demonstrate their goals using direct manipulation of graphical encodings used in visual representations [96]. Previous work also indicated the level of interaction directness with the visual representation contributes towards increasing the sense of control and personal agency in the participants [97]. However, we observed that this level of directness sometimes led the participants to the point that they forget their primary task. For example, one of the participants was so involved in the process of dragging the points that at one point he said: *"I forgot what I was going to do."* This could potentially go against the goal of traditional visualization tools that maintain a functionalist perspective [98], in that they are designed to be helpful for a particular set of analytic tasks. Advantages and disadvantages in increasing the directness of interaction paradigms then raise a question — What is the right level of interaction directness that should be given to the users of the visualization tools?

Reconfiguring a Visualization. We did not find a large difference in the number of times that the participants reconfigured the visualizations using each tool (see Table 5.2). In fact, the results of our first phase also indicate that both tools were quite fast in reconfiguring visualizations. However, we noted four of the participants found sorting the bar chart using VisExemplar intuitive and fun. For instance, one of the participants mentioned: *"[...] the sorting was intuitive."* VisExemplar enables users to demonstrate their interests in sorting the bar chart by dragging the shortest/tallest bar to the extreme left/right. The system then recommends sorting the bar chart. We believe interaction directness in tools implementing VbD affects the feeling of engagement and involvement during visualization process [99, 14]. Another participant said: *"interactions like sorting are fun and natural. Have you ever thought to test your tool on high school students? I think they will like it a lot because they can move things around and play with it while they are learning."*

One interesting avenue of research is to investigate the effectiveness of these interaction paradigms on visualization tools that are designed for different categories of users. For example, while user engagement and involvement might not be the primary goal of tools that are designed to support a particular set of analytic tasks, it might be important for tools that are designed for educational purposes or casual information visualization tools [98].

Switching Between Visualization Techniques. The participants switched between visualization techniques more often while using Polestar (see Table 5.2). The ability to quickly change from one type of technique to another could contribute to this advantage. In particular, seven of the participants found it quite difficult to switch from a scatterplot to a bar chart using VisExemplar. To switch from a scatterplot to a barchart using VisExemplar, the participants had to stack two or more data points vertically. The system then recommended a set of barcharts based on similarity of the data points. Participants found this type of task difficult to demonstrate. For example, one participant expressed the difficulty of switching

from a scatterplot to a barchart by saying: *"it was a bit awkward and hard for me to stack the points to create a barchart."* Without intuitive and easy visual analogies to demonstrate an intended task or goal, the effectiveness of tools implementing VbD may suffer, and other interaction paradigms such as MVS may be better suited.

5.2.7 Discussion

No-need-to-think vs. Need-to-think

To construct different visualizations in Polestar, the participants mapped data attributes to different visual properties through GUI operations visually presented via the control panel. While participants experienced fast visualization construction using Polestar, they generally reflected less on the meaning and potential impact of their interaction. For example, one of the participants stated: *"When I started, I did not have any design in my head. So, I kept creating different designs until I found the one [visualization] that looked interesting."* In such cases participants tried mapping a variety of data attributes to different visual properties until they created a visualization that they liked. Previous work [91] also confirms this notion of "no-need-to-think" when working with Tableau.

In contrast, tools implementing VbD advocate for the idea of constructive visualization, which previous work [38] defined as *"the act of constructing a visualization by assembling blocks, that have previously been assigned a data unit through a mapping."* In tools implementing VbD, users need to think about the visual output or how data mappings should look before starting the demonstration process. With VisExemplar, five participants mentioned that they had to think about how they want their visual outputs to look before starting to provide visual demonstrations to the visual representation. One of the participants stated: *"Here [in VisExemplar] I need to think and imagine the output first. I then need to come up with strategies to show [demonstrate] parts of what I want to the system."* Thus, in contrast to Polestar, VisExemplar advocates for the notion of "need-to-think" before specifying visual properties.

As previous work highlights [89], large amounts of effort and research have gone into designing visualization tools (e.g., Tableau) that enable rapid visualization design through coarse data granularity and little user effort. Such tools are highly valuable for rapid data exploration and visualization construction, and are designed for a broad set of tasks and user expertise. While supporting rapid visualization construction is valuable, we believe that there is value in the continued investigation of alternative tools and interaction paradigms that foster the notion of "need-to-think".

For instance, tools implementing the notion of "need-to-think" such as VisExemplar require users to put more effort into thinking about their data and visualization designs, potentially making the processes of visualization construction slower. However, this notion may lead to a more thoughtful process since the participants often mentioned the need to plan and think about their data prior to engaging in the process of visualization construction. In particular, the notion of "need-to-think" enables users to think more carefully about their data, marks, visual variables, and their relevance in the design and construction of visualizations [91]. As such, optimizing for performance time may not lead to the overall best outcome, as it may result in users glossing over important details of the data, uncertainties, open questions and ultimately decrease insights. As design practices suggest, active involvement and repetition fosters users' creativity and critical thinking [89].

Design Guidelines

DG1: Ease the process of visual demonstration by incorporating more advanced interactions. Providing demonstrations is one of the fundamental steps in the visualization by demonstration paradigm. As such, more advanced interactions can improve the speed and accuracy of the demonstrations provided by users. Going forward, we envision multiple ways to improve user interaction in tools implementing VbD. One way is to incorporate feedforward [100] and suggested interactivity [101] to improve the efficiency and usability. For tasks such as "mapping data attributes to the axes", this could help by showing what

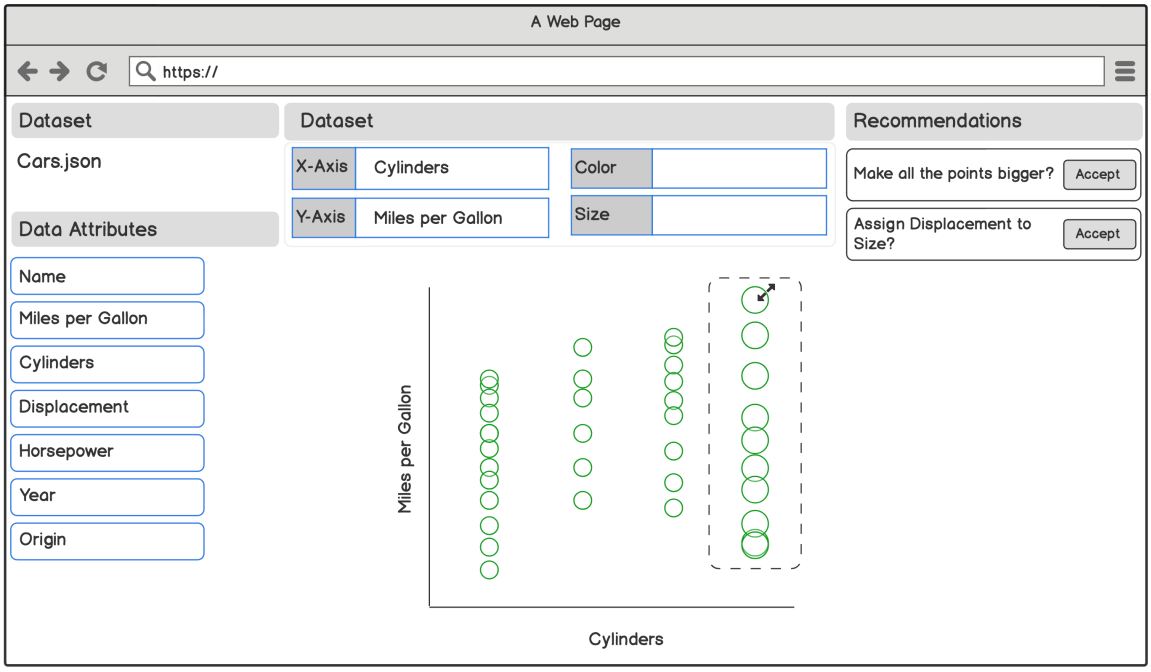


Figure 5.11: Combining MVS and VbD into a single interface opens interesting user interface opportunities that can leverage aspects of both paradigms.

sequences of operations users can execute. Alternatively, interactions such as lasso selection could improve multi-point demonstrations.

DG2: Decrease the ambiguity in user inputs by incorporating multiple input modalities. VisExemplar uses mouse input as the primary form of user interaction. During a visual demonstration, multiple valid interpretations of a user's action can be made. This ambiguity challenge for demonstration-based systems has been previously studied [16], and solutions for disambiguiation exist. While most systems use a fixed model for determining the most "appropriate inference", there is no guaranteed way to either identify the user's intent correctly or be able to resolve the ambiguity without further assistance [16]. In the case of VisExemplar, part of the ambiguity stems from the limited amount of information that direct manipulation of graphical encodings can convey. Going forward, the use of simultaneous modalities (pen, touch, speech, etc.) could decrease ambiguity and increase the amount of information users can provide about their demonstrations.

DG3: Improve Recommendation Interpretation and Timing. While using VisExemplar, participants were sometimes unclear why the system suggested specific recommendations. In such cases, the participants found it difficult to map the recommended options to their interaction with the visualization. Tools implementing VbD suggest potential options based on the given demonstrations. However, there might be cases that the systems do not recommend options expected by the users, and it might not be apparent to users why those recommendations are presented to them. Going forward, we suggest systems implementing VbD to explore design alternatives to explain the reasoning behind recommendations.

We also noticed that the participants sometimes found incoming recommendations interrupting. For example, one of the participants mentioned that *"is there a way to tell the system to do not update the recommendations after each interaction?"* In the current version of VisExemplar, the recommendations will be updated in the interface whenever the recommendation table in the recommendation engine gets updated. We suggest systems which plan to make use of VbD consider investigating methods for minimizing the interruption caused by incoming recommendations. We can envision two such strategies to overcome the timing problem. First, systems present recommendations upon pressing a specific button on the interface. Second, systems could observe the cadence of user interaction with the system and make recommendations at a less active time.

5.2.8 Limitations and Future Work

To compare two interaction paradigms, we had to select two visualization tools that each embody one of the paradigms. Thus, we chose VisExemplar and Polestar because each embodies one of the paradigms and each is relatively simple with regards to the remaining system components. However, user interface design in visualization tools embodying a specific paradigm can be implemented in various ways [102]. For instance, a tool that embodies MVS could be implemented using the shelf configuration, data flow, or visual builder interface design. Each of these implementation variations could influence the con-

struction process differently. iVoLVER [91], for example, follows a data flow-based interface design requiring users to manually draw a visual glyph before binding data to it. While this makes the tool more flexible in terms of customizing the chart, it can also result in the tool potentially being slower than a system like Polestar for specifying standard visualizations. As such, we want to emphasize that the selected tools in this study do not represent all possible interface designs for the MVS and VbD paradigms. Interface design might influence the results of the study. Thus, we encourage future work to consider the effect of tool design when exploring our findings.

We did not control for participants' expertise. We hypothesize that expertise and prior knowledge about visualizations will influence their visualization construction process. For instance, expert users might prefer constructing visualization using the MVS paradigm since they have a better understanding of visual encodings, or because they are familiar using existing tools that leverage MVS. In contrast, novice users might prefer working with tools that embody the VbD paradigm because of its freedom of expression and not requiring users to formalize the mappings between the data and visual encodings. However, this remains to be formally studied.

5.3 How to offer the benefits of both visualization by demonstration and manual view specification in a unified visualization tool?

Both MVS and VbD enable people to iteratively build visualizations and analyze their data. However, MVS and VbD have their own advantages and disadvantages [13]. When learned, MVS tools are fast because they have high external consistency [103, 13]. On the other hand, VbD tools have higher interaction expressivity, thus increase the level of perceived control and engagement for the user [13]. This leads us to consider how complementary these two paradigms are, and if it is possible to leverage the advantages of each, while limiting their respective disadvantages. However, we do not know *how to offer the benefits of two different interaction paradigms in a unified visualization tool.*

Little work has explored how multiple interaction paradigms that use *the same input modality* (mouse for MVS and VbD) can be blended into a single visualization tool. We hypothesize that the expressivity provided by VbD can be a beneficial addition to tools that rely on the well-known MVS paradigm. Combining these two paradigms poses several challenges. First, it requires careful design and implementation considerations to ensure usability and proper combination of the two paradigms. Second, it requires studying empirically the extent to which such an interface facilitates common visual analysis tasks, as well as whether it leads to an improved user experience.

To address these questions, we created , a visual data exploration prototype that unifies the MVS and VbD paradigms. We use as a testbed to investigate opportunities and challenges in combining interaction paradigms. Through the design and implementation of , we exemplify how MVS and VbD can be blended to complement each other. We then report a qualitative study of with 10 participants that shows how people use both interaction paradigms for data exploration. Further, we discuss varying preferences for interaction paradigms, opportunities and challenges in multi-paradigm interfaces.

5.3.1 Preliminary Study

Designing a tool that combines paradigms (MVS and VbD in our case) is a challenging endeavor. We faced several design decisions, including: *Should each operation be supported by only one paradigm (perhaps the one best suited for the operation) or by both paradigms? Should both paradigms work in conjunction or independently?* Thus, we started with a preliminary study to better understand the design space of a visualization tool that combines MVS and VbD.

Our first prototype supported a variety of operations for scatterplots and bar charts, including: mapping data attributes to axes and mark properties (e.g., size and color); switching from one visualization technique to another; filtering data points; and sorting according to an axis. Both MVS and VbD covered all operations. For instance, to sort a bar chart,

Figure 5.12: The interface. The *Show Me* Menu shows the supported visualizations. The *Attribute* Panel lists the attributes in the dataset. The *Filter* Panel shows user-specified filters. Filters are created by dragging and dropping either data attributes or data points onto the panel. The *Recommendation* Panel shows suggestions from the system in response to a demonstrations made by the user. The *Encoding* Panel contains visual encoding placeholders. Users can map data attributes to visual encodings by dragging and dropping attributes onto these placeholders. The *Main View* shows the visualization.

one could either: i) with MVS, click the sort button on the control panel; or ii) with VbD, drag the shortest/tallest bar to extreme left or right – the system then suggested sorting the bar chart.

We recruited four participants (3 male, 1 female). We asked them to imagine their employer had asked them to analyze a dataset about movies (the Movies dataset [68]) using the tool for 20 minutes, and to report their findings about the data. We encouraged participants to try both paradigms and to verbalize their thought process while exploring the data. Results from this preliminary study emphasized three observations for constructing visualizations using a multi-paradigm tool:

Observation 1: Try it out first. At first, participants performed the same operation using both paradigms one after the other. Trying out each paradigm helped them better understand the system and possible interactions.

Observation 2: Assess and choose. Over time, participants converged toward using the paradigm they found to be most efficient for a given operation. For instance, all four participants preferred using MVS to switch between visualizations as they found this easier and more efficient than using VbD. For some operations like filtering data points or mapping a data attribute to color, they found the two paradigms to be equally effective and used both interchangeably.

Observation 3: Combine. Participants sometimes combined MVS and VbD to complete a single operation. For example, a participant first used MVS to map a data attribute to color in a scatterplot, by dragging and dropping the data attribute to the color shelf. Then, she said: *"I don't like the colors"*, and used VbD to manually color a few data points, representative of a custom color palette she had in mind. The system then recommended color mappings based on these colors.

Based on these three observations, we iterated over our initial design to develop a second prototype, called .

5.3.2 Liger Walk-through

We illustrate the functionality of through a scenario. Suppose Amy is interested in buying a car that would best match her needs and preferences. She opens and loads the car dataset [67]. The dataset contains 250 cars with 9 attributes such as number of cylinders and miles per gallon.

Amy first wants to get an idea of how numbers of cylinders relate to miles per gallon. Using the MVS paradigm, she drags the *Cylinders* and *Miles per Gallon* attributes from the *Attribute* panel and drops them onto the x and y axis placeholders in the *Encoding* Panel. She creates her first visualization by selecting the bar chart under the *Show Me* menu.

Amy decides to sort the bar chart to see which numbers of cylinders have the highest and lowest average miles per gallon. For that, she uses the VbD paradigm. She selects the tallest bar and drags it to the right of the bar chart to demonstrate her interest in sorting (see

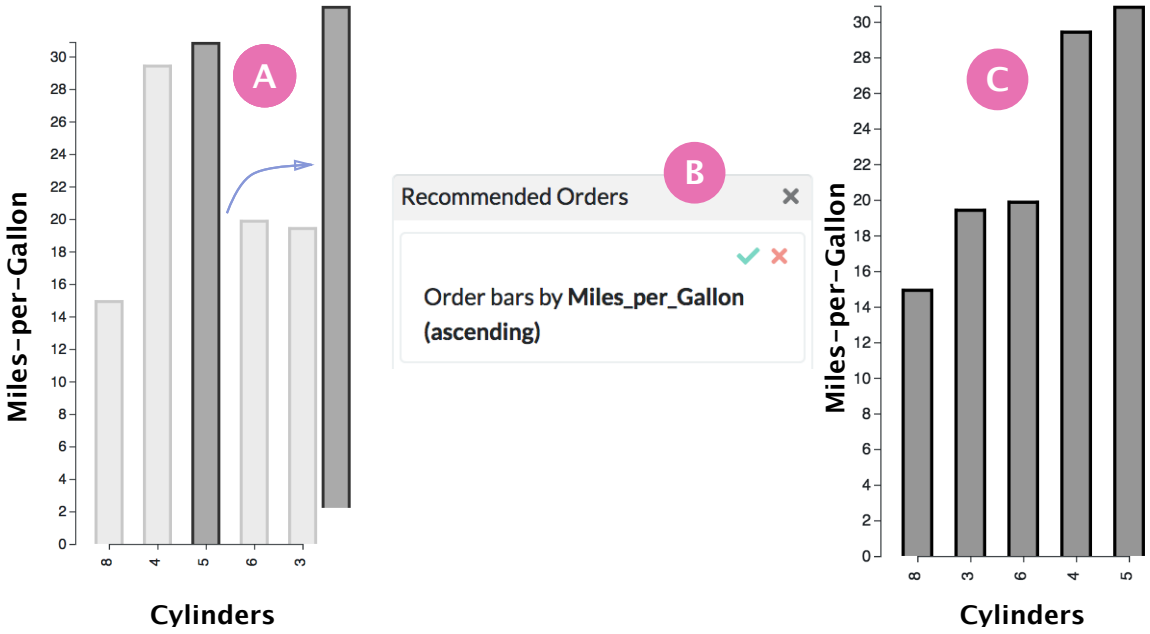


Figure 5.13: After Amy drags the tallest bar to the extreme right of the bar chart **(A)**, the system recommends sorting the bar chart by *Miles Per Gallon* in an ascending order **(B)**. Amy accepts the recommendation to sort by *Miles Per Gallon* **(C)**.

Figure 5.13-A). In response, the *Recommendation Panel* is updated based on the system's interpretation of Amy's demonstration (see Figure 5.13-B). Amy accepts the recommendation to sort the bar chart by *Miles per Gallon* in an ascending order (see Figure 5.13-C).

Amy realizes i) that there is not a straightforward relationship between the number of cylinders and the miles per gallon rating; and ii) that the bar chart is not a good visualization for helping her look at individual vehicles. Thus, she decides to look at the relationships between other dimensions (*Horsepower*, *Acceleration*, and *Cylinders*) using a scatterplot. She uses MVS to switch from the bar chart to a scatterplot. She drags and drops the *Horsepower* and the *Acceleration* attributes onto the x and y axis placeholders, respectively (see Figure 5.14-A). Then, although it is not a very good choice for a quantitative attribute, she maps the *Cylinders* attribute to color hue using the same drag-and-drop technique (see Figure 5.14-B).

Amy is not fond of the color scheme automatically assigned to *Cylinders*. She removes

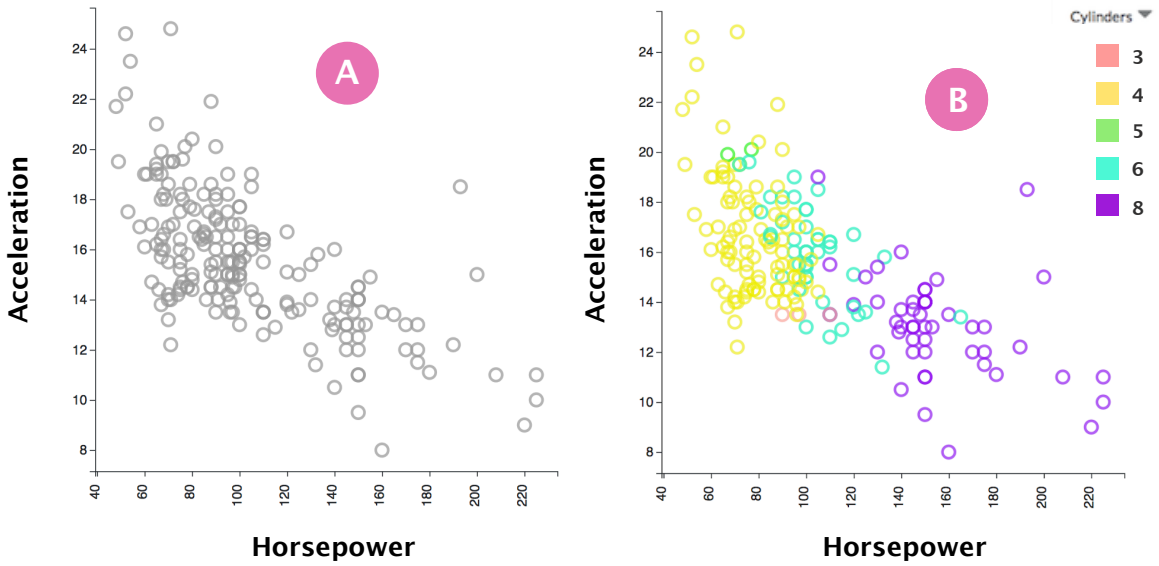


Figure 5.14: Amy creates a scatterplot with *Acceleration* on the x axis and *Horsepower* on the y axis **(A)**, then maps *Cylinder* to color hue **(B)**.

the color mapping (see Figure 5.14-A) then uses VbD to create a color scheme to her taste. To demonstrate her intent to customize the color palette, she selects and re-colors a few 4-cylinder cars red and a few 8-cylinder cars blue (see Figure 5.15-A). The system extracts data attributes that can be mapped to color (in this case *Cylinders* and *Displacement*) and recommends them (see Figure 5.15-B). Amy accepts mapping *Cylinders* to color. The *Encoding* panel now shows the *Cylinders (customized)* attribute on the color placeholder (see Figure 5.15-C) and the color of the data points is updated according to the new color scheme (see Figure 5.15-D).

Amy remembers a friend of her's mentioned that Japanese cars have low fuel consumption. She uses MVS to exclude non-Japanese cars, by dragging and dropping the *Origin* attribute onto the *Filter* Panel (see Figure 5.16-A). The *Filter* panel now shows the three values for the *Origin* attribute. She excludes the European and American cars by deselecting these in the filter (see Figure 5.16-A). This updates the overview of the filtered points in the *Filter* panel and in the *Main View* (Figure 5.16-B).

While low consumption is important, Amy does still wants some sporty aspects to her driving experience, thus seeks a car with high *Horsepower*. She uses VbD to select cars

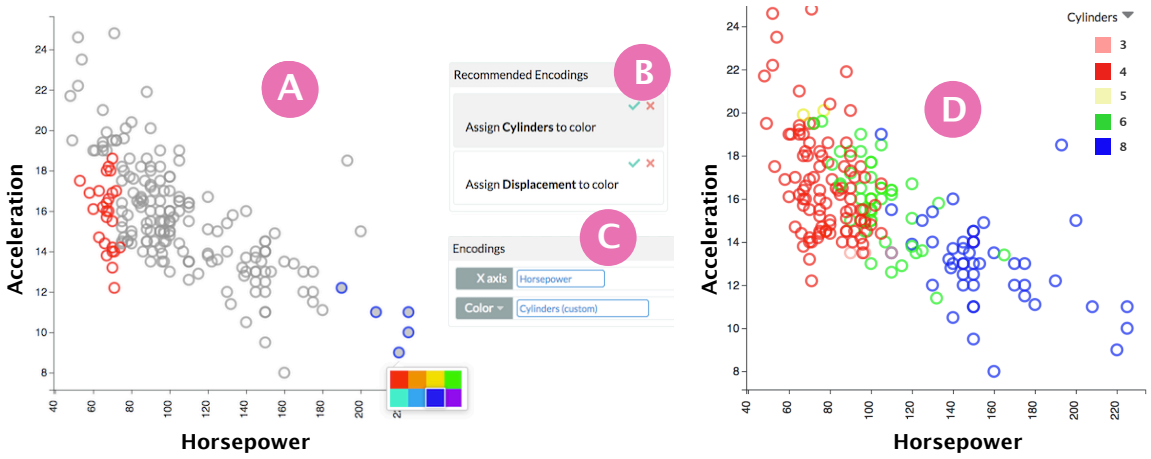


Figure 5.15: Amy colors a few 4-cylinder cars red and a few 8-cylinder cars blue **(A)**. The system recommends assigning either *Cylinder* or *Displacement* to color hue **(B)**. Amy accepts to map *Cylinders* to color. This updates the *Encoding* panel **(C)** as well as the *Main View* **(D)**, using the colors she manually specified.

with *Horsepower* below 100 (see Figure 5.17-A). She demonstrates her interest in filtering out the selected cars by dragging them out of the *Main View* and dropping them onto the *Filter* panel (see Figure 5.17-B). Amy explores the recommended options (see Figure 5.17-C) by hovering over them on the Recommendation Panel, which provides a preview of the change in the main view. She accepts the recommendation to filter out all points within the same x range. The system then automatically creates a new filter for the *Horsepower* attribute in the *Filter* panel. Because the data is quantitative, the filter provides a range slider for Amy to fine-tune the filtering criteria (see Figure 5.17-D). The *Main View* updates to reflect the new filter (see Figure 5.17-E). Amy notices that one of the four cars (in red) has fewer cylinders than the other three (in green). Given that the cars are similar otherwise, she thinks that the **Mazda MX-5** is her best option. After narrowing down her options to only 4 cars, she decides to go test drive each of them.

5.3.3 The Liger Prototype

is a proof of concept prototype that blends MVS and VbD, based on lessons learned from previous work [102, 89, 76, 13, 91] and from our preliminary study. is implemented using

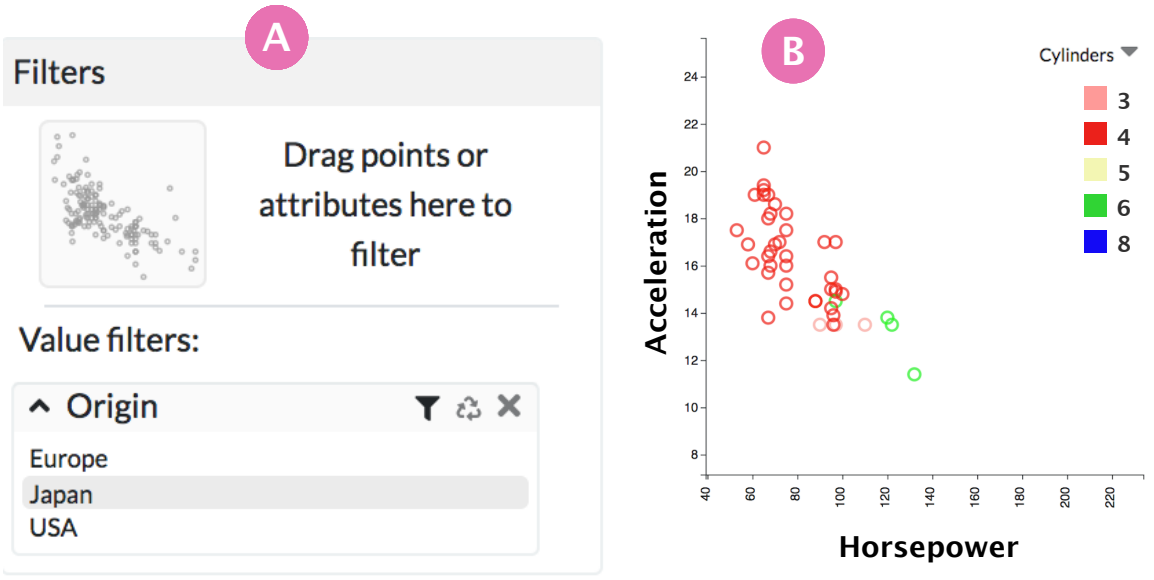


Figure 5.16: Amy creates a filter to filter out European and American cars **(A)**. The *Main View* updates to only show Japanese cars **(B)**.

JavaScript, TypeScript, and D3 [71] and is available at xxxxxxxxxxxxxxxx *[the link is removed for anonymity]*. It currently supports bar chart, stacked bar chart, and scatterplot visualizations, some of the most commonly used visualizations [104]. Next we describe how supports, MVS, VbD, and their combination.

How supports MVS There are two main guidelines for designing MVS tools [88, 28]. First, the tool must support mapping data attributes to various visual encodings. implements this guideline through a shelf-configuration design (i.e., dragging the attributes and dropping them onto encoding shelves in the interface) – similar to Tableau [27]. Second, the tool must update the visualization in real time after visual encodings are created or updated. renders the Main View every time a visualization property is specified.

How supports VbD There are three main guidelines for designing VbD tools [6]. First, the tool must enable direct manipulation of visual representation as a method for providing visual demonstrations. In , the Main View renders the visualization and allows users

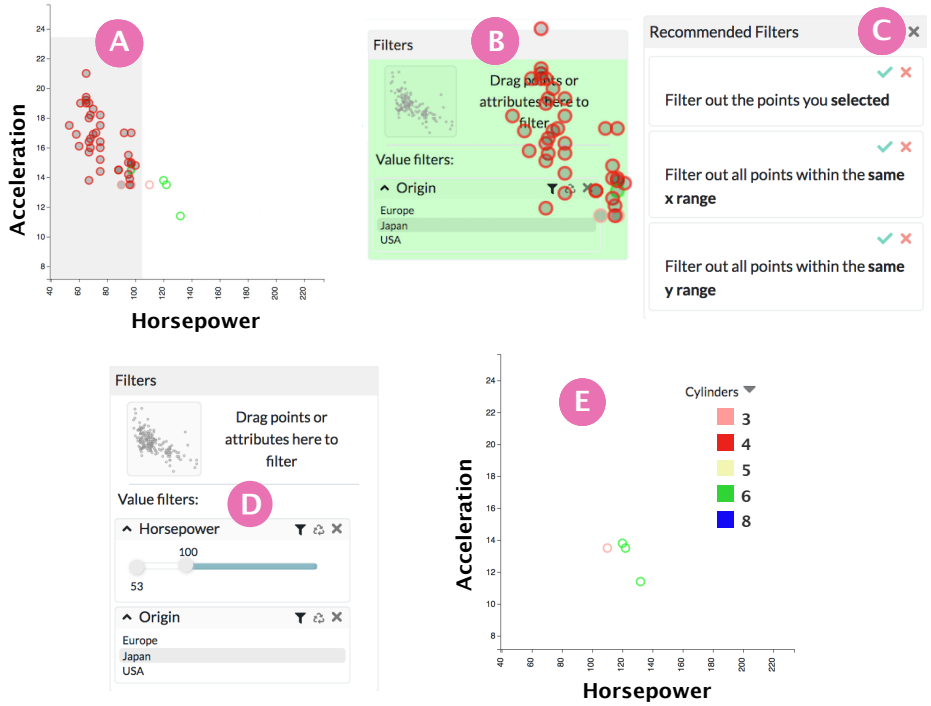


Figure 5.17: Amy draws a rubber-band rectangle to select cars with low *Horsepower* **(A)**. She drags the selected cars and drops them onto the *Filter* panel to demonstrate her interest in filtering out these cars **(B)**. Amy chooses to filter out the cars with *Horsepower* within the selected range. She then uses the range slider to filter out cars with *Horsepower* below 100 **(D)**. Amy ends up with four similar cars to choose from **(E)**.

to provide visual demonstrations by manipulating graphical encodings of the visualization itself (e.g., users can color a few data points to demonstrate their interest in mapping a data attribute to the color encoding). Second, the tool must balance the human and machine workload in the visualization construction process. suggests possible relevant visual transformations in response to given demonstrations (e.g., suggests data attributes that can be mapped to the color encoding). Third, the tool must enhance the interpretability of recommendations. implements this guideline in several ways: i) the Recommendation Panel organizes recommendations in different divisions based on their types (e.g., Recommended Filters, Recommended Encodings); ii) hovering over a recommendation provides a preview (feedforward [100]) of how the visualization would be updated; and iii) each recommendation is explained in natural language.

How combines MVS and VbD Previous work indicates that the effectiveness of interaction paradigms varies depending on the operation at hand [13, 91]. Results from our preliminary study (**Observation 2: Assess and choose**) indicate that people choose a paradigm for a given operation based on its effectiveness for that operation. In , not every operation is supported by both paradigms. Both MVS and VbD support the operations for which they are well-suited.

Results from our preliminary study (**Observation 1: Try out first** and **Observation 3: Combine**) indicate that users appreciate the freedom to switch between paradigms, including times when doing so completes a single operation. allows MVS and VbD to work hand in hand, allowing users to seamlessly switch between paradigms *anytime* during their visualization construction process. For example, one can first create a scatterplot and assign a data attribute to the color of the points using MVS. They can then continue their construction process by mapping a data attribute to the size of the points using VbD. also enable users to switch between paradigms to complete a single operation. For instance, one can first use VbD to filter out a specific set of points, then use a range slider to fine-tune the filtering criteria using MVS.

Results from our preliminary study (**Observation 3: Combine**) indicate that when users perform an operation using one paradigm, the system should show the corollary interactions with the other paradigms. supports this by synchronizing paradigms. For example, if a user maps a data attribute to color using VbD, the system updates the color placeholder on the Encoding Panel to show the equivalent of that interaction using MVS. Or, if a user filters a set of points according to a quantitative data attribute using VbD, the Filter panel shows a range slider that can be further used using MVS.

5.3.4 Operations Supported in Liger

Table 5.3 provides the five operations supported by according to visualization type (bar chart, stacked bar chart, scatterplot) and interaction paradigm (MVS, VbD). Below, we

explain how MVS and VbD support each operation.

: Map Data Attributes to Axes Only MVS supports this operation. Users drag a data attribute from the Attribute Panel and drop it onto one of the shelves on the Encoding Panel.

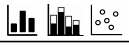
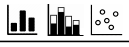
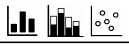
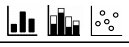
: Map Data Attributes to Mark Properties With MVS, users map a data attribute to color or size by dragging that attribute and dropping it onto the color or size encoding placeholders. With VbD, users first directly manipulate the mark properties for a few data points. Then, recommends data attributes that can be assigned to those properties. For example, in a bar chart, users could color the bars based on their values. In response, the system would recommend potential data attributes that can be mapped to the color encoding.

: Switch Between Visualization Types Only MVS supports this operation, via the "Show Me" menu (top-left menu in Figure 5.12).

: Filter Out Data Points With MVS, users drag a data attribute and drop it onto the Filter Panel. The system then shows the range of filtered values on the Filter Panel. shows the filtered values differently depending on the data attribute type (e.g., quantitative, categorical). For instance, for a quantitative data attribute, the system will show the filtered values through a range slider that can be further tuned. With VbD, users demonstrate their interest in filtering data points by selecting some of those points then dragging and dropping them from the Main View onto the Filter Panel. In response, suggests ways to specify the selection of points to filter.

: Sort the order of Bars With MVS, users change the order of the bars by clicking on the sort bars buttons. These buttons appear on the top menu when the visualization is either a bar chart or a stacked bar chart. With VbD, users demonstrate their interest in sorting a bar chart or a stacked bar chart by dragging the tallest/shortest bar to the extreme left or right side of the visualization. In response, recommends sorting the bars in an ascending or descending order.

Table 5.3: List of operations that supports, according to visualization type and interaction paradigm.

OPERATIONS	VISUALIZATIONS	MVS	VBD
: Map data attributes to axes			
: Map data attributes to mark properties			
: Switch between visualizations			
: Filter out data points			
: Sort the order of bars			

5.3.5 Evaluating Liger

We initially considered performing a study to compare the effectiveness of a multi-paradigm tool with a single-paradigm tool (e.g., Polestar or Tableau). However, our goal is not to examine the benefits of single-paradigm versus multi-paradigm; it is to understand *how* participants use each of the paradigms blended in a unified system, and to study the features and design of . To address this goal, we conducted a think-aloud exploratory observational study, i) to understand participants' processes when **using** (e.g., how often do participants use each interaction paradigm? What types of visualization specifications do participants create using each interaction paradigm?); and ii) to reveal **barriers** of and each paradigm (e.g., when and how difficulties happen). The two datasets used in our study, the operations used for training sessions, the study protocol, and the data we collected are available at xxxxxx *[the link is removed for anonymity]*.

Data Collection Methods

To answer our research questions, we collected a range of data that capture participants' processes and preferences. First, we used questionnaires to collect participant demographic and background information. During the main study, we took written notes of participants' interactions with . We screen- and audio-recorded the whole study. We then conducted a semi-structured interview where we asked participants a set of questions to collect their preferences and subjective opinions about the tool and the two paradigms.

Participants and Settings

We recruited 10 non-color blind participants (2 females, 8 males), aged 22–34 (mean 27.8) via email and word of mouth at our university. None of them had participated in the preliminary study. They were all undergraduate and graduate students who were familiar with reading visualizations, and had created visualizations before. Some participants had used tools such as Microsoft Excel (6), D3.js (3), SPSS (3), Tableau Software (2) and Google Charts (1), and with programming languages such as R (5), Python (4) and Matlab (2).

Datasets

We used two datasets in our study: the Cars dataset for the introduction and training sessions (250 cars, 9 attributes); and the Movies dataset for the main experiment (335 movies, 12 attributes). We selected these datasets because: i) participants were likely to be familiar with the meaning of the attributes (e.g., IMDb rating, profit, genre); and ii) the datasets are complex enough in terms of number of data cases and attributes to support an open-ended data exploration task.

Tasks

We designed 10 *training tasks* that involve the operations supports. For that, we interacted with the tool ourselves and obtained a list of 22 operations (e.g., assign a data attribute to the size or color of data points). Then, using taxonomies of tasks commonly used for interactive visualization construction (e.g., [93, 3, 94, 95]), we assigned each of the 22 operations to one of the five categories listed in Table 5.3. Last, we selected two tasks per category (10 tasks in total), favoring diversity of interactions they involve and coverage of both paradigms. The 10 tasks are detailed in supplemental material. The *main task* is a data exploration task where participants are given and goal then try to achieve this goal using . We opted for an exploratory task because we are interested in the qualitative understanding of how people use a multi-paradigm tool, rather than in measuring the performance of

people to complete low-level tasks accurately and/or quickly.

Procedure

1. Introduction (10 min) We briefed participants about the purpose of the study and their rights. Then we asked them to fill out the study consent form and the questionnaire on demographics and visualization expertise. Next, we gave participants a brief introduction to 's UI where we walked them through different features and supported interactions. We encouraged them to ask questions during this phase.

2. Training (20 min) We gave participants a printed list of the 10 training tasks (2 tasks for each of the 5 categories of operations) to perform on the Cars dataset, in randomized order for each participant. We informed them that we would not measure completion time, so that they interact naturally with the tool and ask as many questions as they want. However, we told participants that they must complete each training task correctly before moving to the next, and that this phase is limited to 15 minutes. Once they had completed the 10 tasks, participants could freely interact with the tool for an additional 5 minutes. Then they took a short break.

3. Main study (20 min) We asked participants to explore the Movies dataset and look for interesting facts about the data. Specifically, we told them: *"Imagine you are planning to watch a movie. Given this dataset about Movies, please make a data-driven decision using our tool for 10–15 minutes and come up with a list of movies that you should be watching."* We asked participants to verbalize questions they have about the data, interactions they perform to answer those questions, and their answers to those questions, in a think-aloud manner. We also asked them to focus on data-driven findings rather than previous knowledge about the data. The participants were could not ask questions during this phase. We did not interrupt the participants except to remind them to think aloud.

4. Follow-up Interview (10 min) After the main study, we asked participants six open-ended questions designed to identify major roadblocks and participants experience with . These questions are available in supplemental material.

5. Wrap-up (5 min) The experimenter thanked the participants, who received a $10 gift card and were invited to ask additional questions about the study.

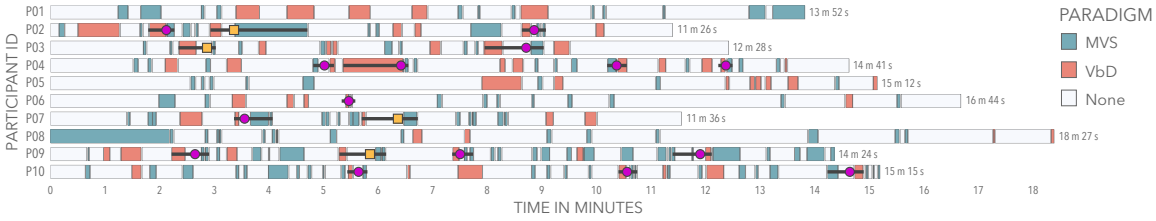


Figure 5.18: The operations participants performed during the *main study* (data exploration). Color indicates which paradigm that was used for each operation. The white spaces indicate when participants were not using any of the paradigms, for instance, when they were hovering over data points or reporting findings about the data. Two symbols indicate when participants switched from one paradigm to the other to perform a single operation, along with a horizontal line that shows the time interval for that operation. A Circle3 indicates that participants combined the two paradigms to perform a single operation; and a rectangle3 that they switched paradigm because they found that their current paradigm was not effective for that operation.

5.3.6 Data Analysis

We analyzed the 251 minutes of screen-capture videos along with the experimenter's notes in three phases.

Phase I A researcher watched 5 random videos out of the 10, to obtain a general sense of the data. Then the researcher coded the paradigm used, visualization type, start time and end time for each operation for all 10 videos (close coding).

Phase II The researcher went through the videos again to identify common and unexpected patterns (open coding) – focusing on participants' *usage* and *barriers*. For example, they looked for cases that participants combined MVS and VbD to complete a single task or cases where participants switched between paradigms because of an inefficiency of one

paradigm for the task at hand.

Phase III A researcher transcribed the interviews. Then they identified meaningful text segments to which they assigned a code word or phrase describing their meaning (open coding). The coding process was iterative with two passes by a single coder in which the coder developed and refined the codes. For example, the codes included phrases such as *"major roadblocks"*, *"strengths"*, and *"combined both paradigms"*. Finally, as a team we identified frequently occurring codes to form higher-level descriptions of the results.

5.3.7 Study Results

We first describe when and how participants used each interaction paradigm. We then explain situations where participants preferred using one paradigm over another.

Do people use multiple paradigms?

As shown in Figure 5.18, **all participants used both interaction paradigms** during the main phase of the study. The interview data reveals that participants found it empowering and effective to be able to leverage both paradigms. For instance, P9 explained how they used both paradigms to filter points differently: *"There were some movies that I did not want to watch or movies similar to them. So, I could simply select them on the plot and drag them out [using VbD]. So, I did not have to look at the panel in that case. The panel [MVS] was also useful, when I knew for example that I wanted to filter the movies with specific IMDB Rating values. So it was giving me that accuracy that I needed in that case. So, I think having both together is useful."* Another participant (P10) talked about the benefits of being able to switch between the paradigms on demand: *"Combining two [paradigms] helps a lot with giving a lot of user control. Like I could do it whatever way I prefer to do it. So, if doing one thing in a specific way [paradigm] is not super natural to me then I can do it another way."*

Several participants **combined the two paradigms** to perform a single task. For in-

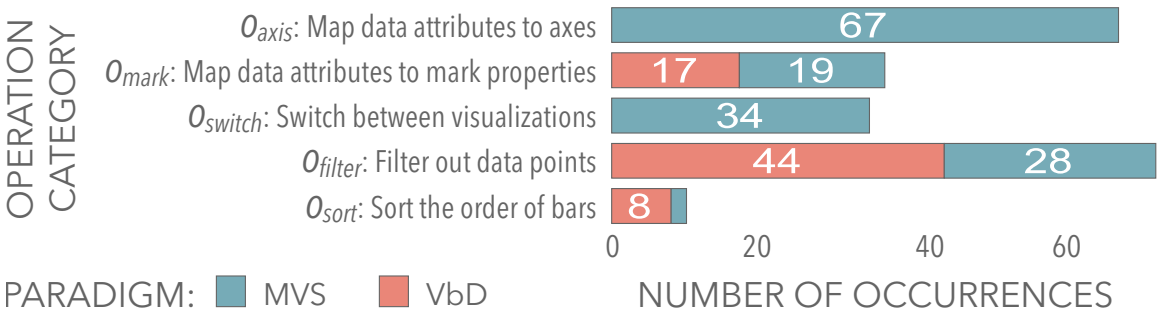


Figure 5.19: The number of times participants performed each operation using each paradigm.

stance, participants combined VbD and MVS 12 times to perform . Many times, they first filtered out a subset of data points by dragging and dropping them onto the Filter Panel (VbD). In response, the system recommended different filtering options. After accepting a recommendation, participants continued their operation by using the range slider to fine tune their filtering criteria (MVS). For example, P2 said during the interview: *"I used two techniques [paradigms] for filtering. Because the demonstration filtering is intuitive but not very precise. So, for precision I fine tuned it using slider."* P9 also mentioned: *"I prefer to do it by demonstration but it is not always very accurate so I had to use sliders on this panel [Filter panel] to get the exact values."* In another example, participants first used MVS to map a data attribute to color encoding (), by dragging and dropping the data attribute to the color shelf. Then they colored a few data points using VbD to indicate their interest in customizing the color palette. The system then recommended color mappings containing the specified colors.

We also noticed that participants sometimes **switched between paradigms** because they found one paradigm less effective for a given operation. For example, three participants switched from VbD to MVS to perform . When we asked participants to explain why they switched to MVS, they mentioned that the system did not recommend what they expected. For example, P2 said: *"when I wanted to assign color to directors, doing it by demonstration I did not get a recommendation that I wanted. I then said let's do it using another technique [MVS]."* P3 also noted: *"for cases that suggestions were not accurate I*

preferred using drag and drop [MVS].”

Which interaction paradigm do people use more often?

Figure 5.19 shows the number of times participants performed each operation with each paradigm. Participants performed 150 operations using MVS and 69 using VbD in total – including and , the operations that only MVS supports. Participants found MVS effective for performing and and particularly liked how MVS is consistent for mapping data attributes to different visual encodings. For all operations that both MVS and VbD support (, ,), **participants used both paradigms in similar proportions** (50 occurrences using MVS and 69 using VbD).

During our interviews, we asked participants to explain the situations where they found one interaction paradigm more effective/useful than another one. 8/10 participants found MVS easy to learn and effective for most operations. For example, P4 said: *“I used the dragging [MVS] in many cases because I knew what exactly what the outcome would be.”* P6 also stated: *“I preferred using drag and drop over demonstration for many tasks because it was easier to use.”*

Most participants (9/10) found VbD more effective/useful than MVS for , for several reasons. P2 found performing with VbD intuitive: *“I definitely like the filtering by demonstration. Being able to highlight parts and drag them out was I think very intuitive.”* P9 found the interaction more user-friendly than with MVS: *“I use Tableau to analyze data in my research. I don’t know if filtering has been this user friendly in Tableau.”* P7 liked how VbD makes it easier to filter attributes with large cardinalities such as movie directors: *“The drag and drop road [MVS] for filtering attributes that have a large number of categorical variables [...] is really hard. So in that case the demonstration road is much easier. I can select instances of the movies that I want to remove on the visualization and drag them out and the system recommends me options for filtering out.”*

Participants had different takes on which paradigm to use for . P3, P5, and P7 men-

tioned preferred using MVS to map data attributes to size and color. For example, P3 said *"I don't like much about the coloring or resizing using demonstration approach, I would rather just drag the variable to the color [MVS]"*, and P5 said *"I feel dragging variables to color is easier than using demonstration" (P5).* Conversely, P8 and P9 preferred VbD for this operation. For example, P8 said *"I love this feature because it allows me to color them directly instead of me thinking which attribute should I assign to color, so that these points have the same color. [...] the system then takes attributes that are similar between those points and suggest me attributes. That is the beauty of this system."* P2 had a unique take on , stating that his choice of paradigm for this operation relied on the size of the dataset: *"it depends on how much data we are dealing. With a small dataset I prefer doing the coloring by demonstration, but with the larger dataset, I would probably say no let's do drag and drop."*

P3, P5, P7, P9 and P10 used VbD to sort bars (), and only P8 used MVS to sort bars (twice). Only P10 commented on this operation, saying: *"dragging the bars to sort the bar chart is good. In general it feels very natural to do."* (VbD).

5.3.8 Discussion

Results from our qualitative study highlight challenges, possible solutions, and future directions for multi-paradigm interfaces. Specifically, we discuss the benefits and drawbacks of being able to choose among interaction paradigms, the cost associated with switching between paradigms, the challenges of learnability and discoverability of multi-paradigm tools, and the challenges in combining paradigms. Further, we discuss limitations and avenues for future work informed by our experience designing and studying a multi-paradigm tool.

Choosing among Interaction Paradigms

All participants in our study did use both paradigms and found the tool empowering and effective. However, because multi-paradigm tools like require people to choose among paradigms while forming a goal, they are likely to increase users' **cognitive load**. Participants sometimes paused for a few seconds to decide which paradigm to choose for performing the operation at hand. For example, while mapping a data attribute to color, P1 said: *"Let me see. hmm. I will go with drag and drop."* In another example, before filtering out some data points, P6 stated: *"not sure which approach to use here."* These observations echo previous findings that even though humans may want freedom of choice and flexibility, making choices requires mental and visual concentration [105, 106].

More generally, multi-paradigm tools might require users to put more effort into thinking about the data and the task at hand, and selecting an interaction paradigm to perform that task. This is likely to make the process of visualization construction **slower** in such tools. However, a visualization is not just a means to an end. Reflection on the data, tasks at hand, and possible interactions also take place during data exploration [107]. Optimizing for efficiency may not lead to the overall best outcome, as it may result in users glossing over important details of the data, the operations, and how to best perform the operations. Indeed, "slow" data exploration can results in users' active involvement, foster creativity and critical thinking, and encourage conscious, deliberate, analytical reasoning [108, 109]. We hypothesize that multi-paradigm tools can bolster deliberate and more logical processes by requiring users to think carefully about their tasks at hand and the different ways of achieving these tasks.

Interaction Cost in Switching Between Paradigms

Switching between paradigms likely increases **interaction cost** [110]. When using a paradigm over a certain period of time, people form habits and learn the design principles that drive this paradigm [111]. After forming a habit with one paradigm, switching to another

paradigm requires breaking this habit and spending some time to recall the principles of the new paradigm. This point is supported by our study findings as participants sometimes felt confused about the functionality of a paradigm after switching to it. For instance, after P5 had used MVS to drag and drop data attributes to the x and y axis shelves, he said *"let me color using demonstration"*, then selected a subset of points and dragged them to the color shelf. He then immediately said: *"Oh. I can't do that."*

Decreasing the interaction cost that results from switching between paradigms is a challenge for multi-paradigm visualization tools. One solution is to design mechanisms such as feedforward [111, 100] to communicate what can be done with the paradigm that is being used, as well as feedback [111] to recall design principles of the other paradigm(s).

Teaching Multi-paradigm Interfaces in Context

Combining multiple paradigms can increase the number of functionality supported by a tool. We know that people should discover and learn the functionality supported by each paradigm as they use the tool over time [112]. However, discoverability and learnability of interaction still is an important challenge to address in visualization [99, 113]. Researching ways of improving **discoverability** and **learnability** of multi-paradigm tools is a promising avenue for future work.

One way to overcome this challenge is to design mechanisms to teach multi-paradigm interfaces in context. When a user performs an operation using a paradigm, the system can teach her how to perform that operation using another paradigm. For example, upon using VbD to map an attribute to color, the system could present how the same operation can be achieved using MVS (e.g., by showing an animation where the mapped attribute moves to the color shelf on the Encoding Panel). Exploring the large design space of techniques for teaching multi-paradigms in context, and assessing the effectiveness of such techniques, is a promising avenue for future research.

There also exists design guidelines on how to enhance learnability and discoverabil-

ity in multimodal interfaces [114, 115]. Because multi-paradigm and multi-modal interfaces share similarities, investigating the extent to which these guidelines apply to multi-paradigm interfaces is promising. For example, we envision displaying contextually relevant interaction options on the interface as users interact with a tool [114]. This could be achieved by providing recommendations that show the available interactions for the next steps of data exploration.

Challenges in Combining MVS and VbD in Liger

Below, we describe challenges that we encountered during the design process of .

From our preliminary study we learned that not every operation must be supported by every paradigm. However, our main challenge was to **identify a set of operations that need to be supported by each as well as both paradigms**. To design , we relied on the results of our preliminary study and on previous work that measured the effectiveness of different operations using MVS and VbD [13]. In future implementations of multi-paradigm tools, we will need to first study the effectiveness of each paradigm for each operation.

Another challenge that we encountered when trying to enable paradigms to work in conjunction was to take into account screen real estate and keep the interface less occupied with interface elements such as menus and shelves. We approached this challenge by designing **interface elements that are shared between paradigms**. For example, the Filter Panel is shared between both paradigms in . Users could drag-and-drop data attributes (using MVS) or data points (using VbD) onto the Filter Panel to complete a filtering operation. Sharing interface elements not only enables us to better use the screen real estate, but also enables us to move toward the ultimate goal of *blending* paradigms rather than just supporting two independent paradigms.

Facilitating synchronization between paradigms also required us to understand how to incorporate changes made by one paradigm into subsequent actions with another paradigm.

To address this challenge, we **transferred knowledge between paradigms** by leveraging user interaction with one paradigm to another. For example, if a user creates a customized color mapping using VbD, the next time the user assigns the same attribute to the color encoding the system preserves the color palette specified by the user.

Limitations and Future Work

To investigate the challenge of blending MVS and VbD paradigms in a visualization tool, we selected two specific interface designs for the MVS and VbD paradigms. However, user interface design in visualization tools embodying a specific paradigm can be implemented in various ways [102]. For instance, a tool that embodies MVS could be implemented using the shelf configuration, data flow, or visual builder interface design. Each of these implementation variations influences the visualization construction process. As such, we emphasize that the selected designs in this study do not represent all possible interface designs for the MVS and VbD paradigms, and that our results unlikely apply fully to all possible designs. Thus, we encourage future work to consider the effect of tool design when building on our findings.

CHAPTER 6
VISUAL DATA CLUSTERING BY DEMONSTRATION

RQ7: *How can visualization by demonstration enable domain experts to perform real world tasks such as data clustering?*

6.1 How can visualization by demonstration enable domain experts to perform real world tasks such as data clustering?

Clustering is the task of summarizing and aggregating complex multi-dimensional data in such a way that items in the same group are more similar to each other than those in different groups. Domain experts often want to perform clustering to find groups of data items that share common characteristics with respect to data attributes. For example, a biologist who wants to investigate genome data can cluster gene sequential data according to similarity between their expression profiles. Clustering has a widespread application in several domains [116, 117, 118].

We aim to accommodate the process of interactive visual clustering for biologists. Like other domain experts, biologists also want to cluster their data and visualize the result to investigate patterns, relationships, and structures among data instances and attributes. However, not all biologists often have formal data science training. The lack of knowledge in data science often prevents users from clustering their data and from interpreting the results in the biological context using the existing tools. Based on our collaborations with a group of biologists, we found that they use tools such as SAS and/or programming languages like R to run cluster analysis on their data. These tools require users to specify clustering algorithms and parameters in written scripts. The absence of user-friendly tools may increase execution costs and impede the adoption of clustering methods for data exploration.

There is a large body of visual analytic systems that employ visual clustering as a part of high dimensional data analysis (e.g., [119, 120, 121, 122, 123, 124, 125]). Some of these visual analytic systems are often complex, and require careful tuning, steering, and parameterization of the clustering models. Interaction complexity in such systems often poses fundamental usability challenges for those domain experts who may not have formal data science training [126]. Furthermore, it is challenging for domain experts to directly apply their knowledge into the clustering processes. For example, biologists exploring genome data might want to merge two clusters because of the similarity of evolutionary history of the genes located in two clusters. Alternatively, they might want to subdivide a specific cluster to estimate the disease risk of genes in different sub-clusters in a specific population. As such, current tools are ill-equipped to help biologists build and explore alternate groupings based on their domain expertise, hindering their ability to discover patterns in the data. Many such tools lack usable interactions to allow domain experts to translate domain-specific questions and hypotheses about the data into model parameters to foster the exploratory process of their tasks.

To tackle the challenges for biologists, we present **Geono-Cluster**, a visualization tool that applies the "by demonstration" [6] paradigm. Instead of requiring biologists to transform their clustering tasks into system specifications by going through layers of menus or programming it, Geono-Cluster allows biologists to directly apply their domain expertise by visually demonstrating how their expected changes should look like (e.g., dragging one cluster and dropping it over another cluster to show their interest in merging the clusters). By translating these demonstrations into numerical processes that update the underlying cluster distance functions, the system predicts biologists' intentions and generates potential clustering results (e.g., different visual clustering outputs that merged those two clusters). We have developed Geono-Cluster in collaboration with biologists investigating disease risks frequency across different populations. We closely followed the design study protocol [127] to derive system requirements, tasks to be supported, and design guidelines based

on feedback from biologists.

We conducted a qualitative study with six expert biologists. In this evaluation, we observed how our tool helps biologists to cluster their data and identify challenges they encounter while using our tool. We also conducted a semi-structured interview to collect biologists' feedback and new ideas. Our results demonstrate that Geono-Cluster enables biologists to build, refine, and evaluate clustering outcomes with intuitive demonstration-based interaction and to interactively explore the results through multiple views.

6.1.1 Formative Assessment

Here we explain a formative assessment that we conducted to characterize users' workflow, derive tasks and requirements from it, to generate design guidelines to design the system.

Characterizing domain experts, data, and tasks

The motivation of this work stems from an ongoing project in which we have been collaborating with biologists at the Georgia Tech. We have been working with the biologists over the past 13 months to design and build solutions for supporting interactive visual clustering of disease risk factors.

The dataset used by the biologists is from Genome Wide Association Studies (GWAS Catalog) [128] which includes published SNPs (single-nucleotide polymorphisms, representing differences in a single DNA building block, called a nucleotide), and association studies to analyse genetic sequences. Through this dataset, biologists intend to determine "alleles" that correlate to various diseases and traits. Alleles are various forms of a gene that are formed by mutation and are found at the same place on a chromosome. Using GWAS dataset biologists analyse SNPs to find how do they vary between various genome samples.

During data analysis, biologists often focus on certain features of their dataset such as, disease/trait, SNP identification number, risk allele frequency, p-value, and odds ra-

tio/beta. Focusing on those values, they try to answer questions like, how and why disease risk frequencies differ across populations, what are the statistical power to detect those known SNPs, and how well associations found in one population can transfer/replicate well to another population. To answer such questions, researchers cluster their data to investigate patterns and relationships of position on the genome, risk allele, and risk allele frequencies that impact diseases risk frequencies across different populations. This is an iterative process and biologists frequently create customized clusters, merge/split clusters, and investigate sub-clusters within a specific cluster to test their hypotheses based on their expertise.

To cluster and visualize their data, these biologists currently use tools/programming languages like R [129] and SAS [130]. They revealed that the current process of clustering and visualizing the data is rather time-consuming, cumbersome, and occasionally error-prone. Our observations as well as researchers' feedback show that they often need to write and execute scripts. Writing scripts becomes even more challenging when they want to perform more specific tasks such as merging two or more clusters. Moreover, the interactive visual clustering tools are ill-equipped to support specific and customized tasks often performed by these biologists. To overcome the challenges, we aimed to design an interactive visual clustering tool that enables visual data clustering for biologists.

Tasks and Requirements

Following a user-centered method [112], we began our iterative design process by investigating current practices, needs, and challenges. We conducted multiple group discussions with two biologists at the Georgia Tech. We started our discussions with the biologists by asking them: 1) what kinds of questions do they ask and answer while exploring their data? 2) why do they perform clustering tasks during their analysis?, and 3) how do they currently create clusters? Then, we freely continued our conversation that touched upon the tools, analytic methods, and challenges they face during the process. We took notes during all

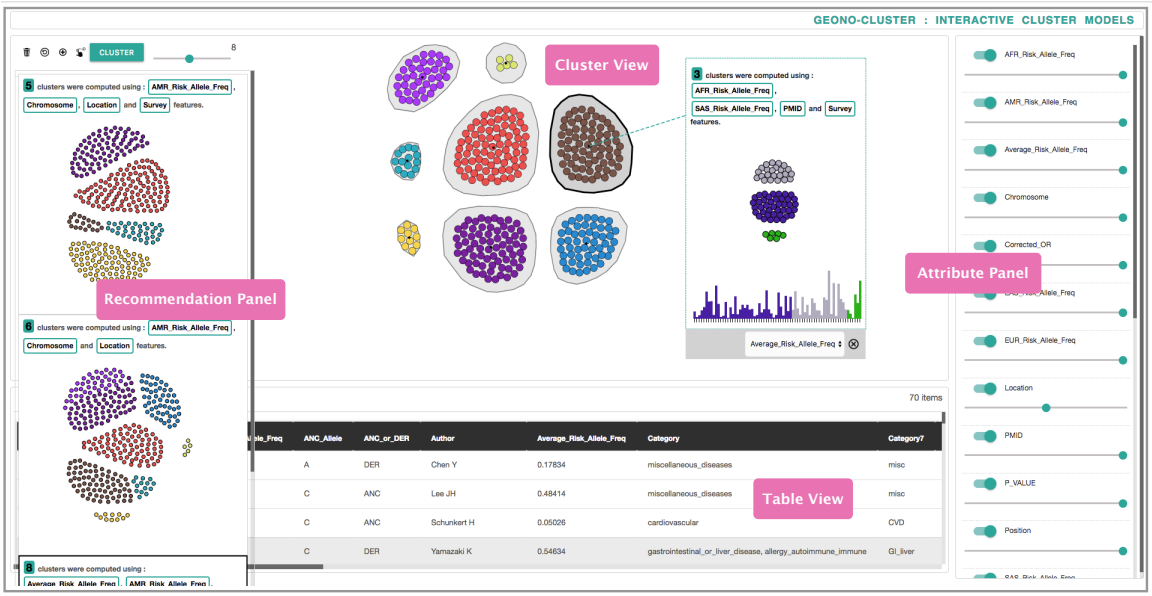


Figure 6.1: The Geono-Cluster user interface consists of a Cluster View, a Recommendation Panel, a Table View, and an Attribute Panel. Cluster view visualizes the clustered data and provide a medium for users to provide visual demonstrations. Recommendation panel shows different clustering results based on the demonstrations provided by users. Table view shows a tabular representation of the loaded dataset. Attribute Panel lists the attributes of the loaded dataset and their weights.

the group discussions. We then read through our notes to gain a better understanding of the requirements and challenges these biologists encounter while clustering their data. After reading the data, we identified the meaningful text segments (e.g., *"[...] here we combine these two clusters."*). We then assigned a code phrase that describes the meaning of the text segment (e.g., merging clusters).

We initially identified three commonly performed clustering operations that are currently challenging for biologists to complete using existing programming languages and tools. Here we define "clusters" or "clustering" interchangeably in two contexts: (1) Algorithmic clustering models such as K-Means,etc, and (2) Group of data items assigned to a collection based on an algorithmic cluster model represented visually as a grouped node view.

T1: Hand-craft, Merge, and Split Clusters: Biologists apply their domain knowledge to create customized clusters to better understand which factor(s) is causing the ascertain-

ment bias on the dataset that are being used popularly. For example, one of the biologist stated: *"Given the identified SNPs [single-nucleotide polymorphisms] that are associated with common disease and traits, it's interesting to create a cluster of SNPs."* In addition, biologists apply their domain expertise to merge or split two or more clusters depending on how related they think the clusters are based on given feature(s). For example, one of the biologists mentioned: *"Depending on the evolutionary history of the genes, two or more clusters can be really related to each other. If ascertained they are related, we will merge them as one cluster."* Another biologist reported that *"In my new project, we are comparing Africans to non-Africans. In this case I merge Americans, East Asians, and Europeans as one cluster, and compare that to Africans data."*

T2: Divide each cluster to sub-clusters: Biologists often investigate sub-clusters within a specific cluster to: 1) understand which other factors can affect the cluster, 2) compare two clusters based on the member data items in each, and 3) see trends and patterns in the sub-clusters, with respect to chosen features, We noticed that the biologists found existing solutions challenging because they had to write lines of scripts to compute and visualize sub-clusters in a given cluster. Furthermore, the existing methods prohibit rapid iteration and visualization of results, which inevitably prolongs the exploratory clustering process to understand their data better.

T3: Adjust feature contributions: Biologists need to easily see by how much different attributes/features contribute to computing a cluster. Moreover, they often need to adjust the importance of different features used for computing a cluster. Biologists currently have to programmatically adjust the importance of features, execute the code, and visualize the outcome. They often repeat this process multiple times until they achieve a satisfactory result. They need interactive methods to view and refine feature contributions.

6.1.2 Design Guidelines

We needed to explore alternatives and make design decisions to better support the aforementioned tasks. In particular, Geono-Cluster should be easy to use by experts who do not have formal data science training. We developed a set of design guidelines to inform those interested in developing visual analytic tools for domain experts (in particular biologists). These guidelines are based on existing tools designed for supporting visual data exploration for biologists [131, 132], mixed-initiative systems [133], and our experiences through several design iterations with biologists.

G1: Shifting the burden of specification from the biologists to the systems. The existing tools and technologies put the burden of specification on biologists[129, 134, 130, 135]. For instance, while clustering their data, biologists need to specify the clustering algorithm, number of clusters and iterations, and other parameters. Our initial interviews show that biologists find the current process time-consuming and cumbersome. For instance, one of the biologists noted: *"It sometimes takes time to perform specific tasks [using Python]. I have to Google and find out how to do it."* Instead of requiring biologists to specify the clustering models by programming or going through layers of menus, the tool should provide an environment that enables them to demonstrate how the expected clustering outcomes should look like [6]. By translating the given demonstrations, the system could estimate the biologist's intention and generate appropriate results. This way we could balance the responsibility between the biologist and the system – biologists provide visual demonstrations, based on this, the system infers potential clustering results and recommends them.

G2: Enable user interaction to drive recommendations. As analysts explore their data, their interests will evolve [5]. Our initial observations and interviews also showed that biologists need to explore various clustering models rapidly during their data analysis process. One potential approach to support such a rapid data analysis is to recommend potential cluster models that biologists should consider during their data analysis process [133,

132]. Furthermore, the clustering recommendations should be adapted for biologists' analytic goals. The recommendation engine should steer multiple clustering models based on biologist-specified expected visual outcomes. In addition, biologists can also directly adjust feature contributions to update the clustering results. In aggregate, these interactions create demonstrations which serve as the primary units by which biologists communicate their expected changes to the system.

G3: Enhance interpretability of recommendations. Biologists reported their interest in seeing more details about different clustering results while skimming through different recommendations. However, not all biologists might be familiar with technical terms used to describe a cluster such as silhouette value. Therefore, recommended clustering results should be presented in a transparent manner so that biologists can extract the most important and understandable information (e.g., contributing features) used for clustering results. One powerful approach to enhance transparency of the recommended clustering options is to use natural language [136] to explain them. This way biologists can learn about the recommended clustering outcomes without having to know about more technical terms describing each clustering outcome.

6.1.3 Geono-Cluster

Based on the tasks and guidelines, we developed Geono-Cluster, a visual clustering tool for biologists. All components of the Geono-Cluster were implemented using JavaScript, D3.js, and Python.

6.1.4 Usage Scenario

In this section, we motivate the design of our system and illustrate the functionality via a usage scenario. We indicate how a domain expert can utilize Geono-Cluster to perform visual cluster analysis on the GWAS Catalog dataset [128]. This dataset includes detailed information regarding the identified single-nucleotide polymorphisms (SNPs) associated

with common diseases and traits (e.g., position on the genome, risk allele frequencies, p-value, effect sizes, etc.). SNP is a region on the gene where more than one allele (A, C, G, T) is observed and each row on the dataset is a SNP [137].

Megan is a biologist who wants to compare populations from the *GWAS* dataset to understand disease risk factors related to geographical regions (e.g., if gene samples collected from "America" are more prone to cancer than gene samples collected from "Europe"). She launches Geono-Cluster to cluster the data, and to compare associated sub-populations. First, Megan skims through different features on the Table View (see Figure 6.1).

Megan knows that there are two types of gene samples: ANC and DER. ANC samples are the genes that are derived from either humans or monkeys. DER are the gene samples that are derived from the mixture of humans and monkeys. Megan starts her exploration by comparing the disease risk factor between the two types of genes samples based on their ancestry. Megan first skims through different features to find the *ANC-or-DER* feature on the Table View. She clicks on a cell in the column *ANC-or-DER* with the value *ANC* in the Table View to demonstrate her interest in selecting all the data items with ancestry *ANC*. In response, Geono-Cluster automatically selects all data items with ancestry *ANC* (see Figure 6.2-A). Megan then demonstrates her interest in clustering data items with *ANC* value by dragging them from the Table View and dropping them to the Cluster View. In response to the demonstration, the system automatically represents data items as red circles and places them in the *Red Cluster* (see Figure 6.2-B). At this point, the system also recommends potential clustering results based on the demonstration provided by Megan (see Figure 6.2-C). Even though Megan's interactions may lead to grouping the data based on the chosen categorical data attribute (*ANC or DER*), in essence, this is a start to allow a user demonstrate their intent to find a clustering model that represents agreeable clusters in the data. Their interactions are inferred as implicit intents by the system to find the most appropriate cluster model as opposed to just group the data by a set of categorical variables.

Megan opens the recommendation panel and previews other clustering options through

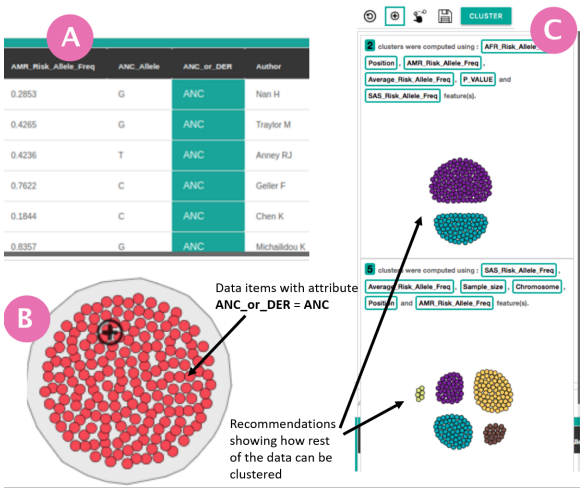


Figure 6.2: **A)** Megan clicks on a cell in the column *ANC-or-DER* with the value *ANC*. The system automatically selects all data items with ancestory *ANC*. **B)** She drags the selected data items and drops them to the cluster view. The system automatically represents data items as red circles and places them in an independent cluster. **C)** The system also recommends potential clustering layouts of the non-interacted data instances based on the demonstration provided by Megan.

the thumbnail previews. She finds one of the clustering results recommended by the system interesting. She clicks on this thumbnail (the first recommendation), which updates the Cluster View with the recommended cluster layout by adding the *Blue Cluster* and the *Purple Cluster* (along with the *Red Cluster*) in the Cluster View (see Figure 6.3-A).

Megan explores the data items within each cluster by hovering over each data item to see its details. She notices that while the *Red Cluster* contains genome samples with ancestry *ANC*, the recently added clusters (*Blue Cluster* and the *Purple Cluster*) contains all the gene samples with ancestry *DER*. She further notices that most of the items in the *Blue Cluster* have the chromosome value higher than 8, and the genome samples belong to the region *America*. Now she understands what each clusters represent.

Next, Megan demonstrates her interest of excluding data items with ancestry *ANC* that belong to *Africa* from the *Red Cluster*. To do so, she lasso-selects a subset of data items with ancestry *ANC* that belong to the region *Africa* from the *Red Cluster* (see Figure 6.3-A). Important to note that points closer to each other in a cluster are expected to be similar to one another based on the applied cluster model. Thus using the lasso selection, Megan selects similar points from a cluster for further analysis. She then drag-and-drops these

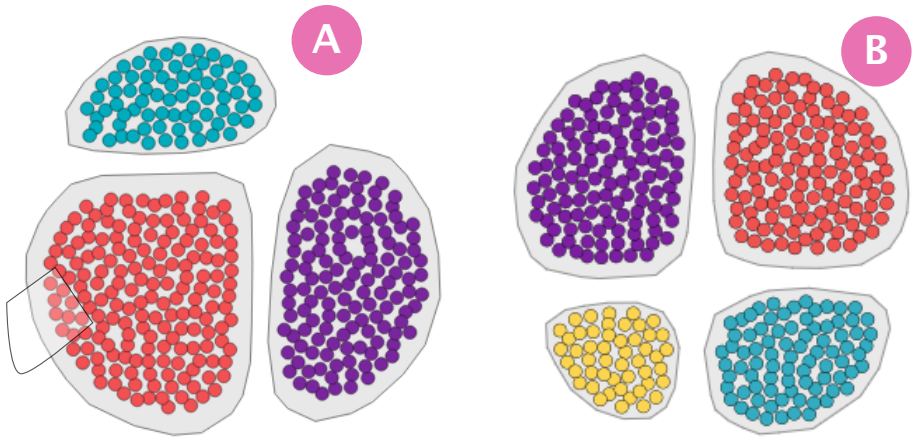


Figure 6.3: **A)** Megan uses lasso tool to select a subset of data items from the red cluster and drags them out. **B** The system automatically finds other similar data items and defines the yellow cluster containing them.

points out of the cluster. In response, the system automatically finds other similar data items with ancestry *ANC* that belong to the region *Africa*, and then defines the *Yellow Cluster* containing these data items.Further, the system updates the recommendations in the recommendation panel accordingly.

Looking at the *Purple Cluster*, and the *Yellow Cluster*, Megan realizes that the items in these two cluster are with ancestry *ANC* and *DER* respectively. Megan wants to compare the distribution of the feature *Average-Risk-allele* between these two clusters to compare their disease risk factors. She clicks on the "+" icon (see Figure 6.4), which is shown upon hovering on a cluster, to open the sub-cluster panel for each cluster. Each sub-cluster further clusters the data items per cluster. Also, the sub-cluster panel contains a bar chart, highlighting the distribution of a chosen feature (*Average-Risk-allele*) through a drop-down selector for all data items in the parent cluster (see Figure 6.4). After inspecting the distributions, Megan does not notice any significant difference in the value of *Average-Risk-Allele* between the *Purple* and the *Yellow* cluster.

Megan explores the *Blue Cluster* (with *DER* ancestory) to inspect its sub-cluster layout and distribution of the feature *Average-Risk-Allele*. When she compares the distribution of feature contributions of the *Yellow Cluster*, she discovers that the genes sampled from

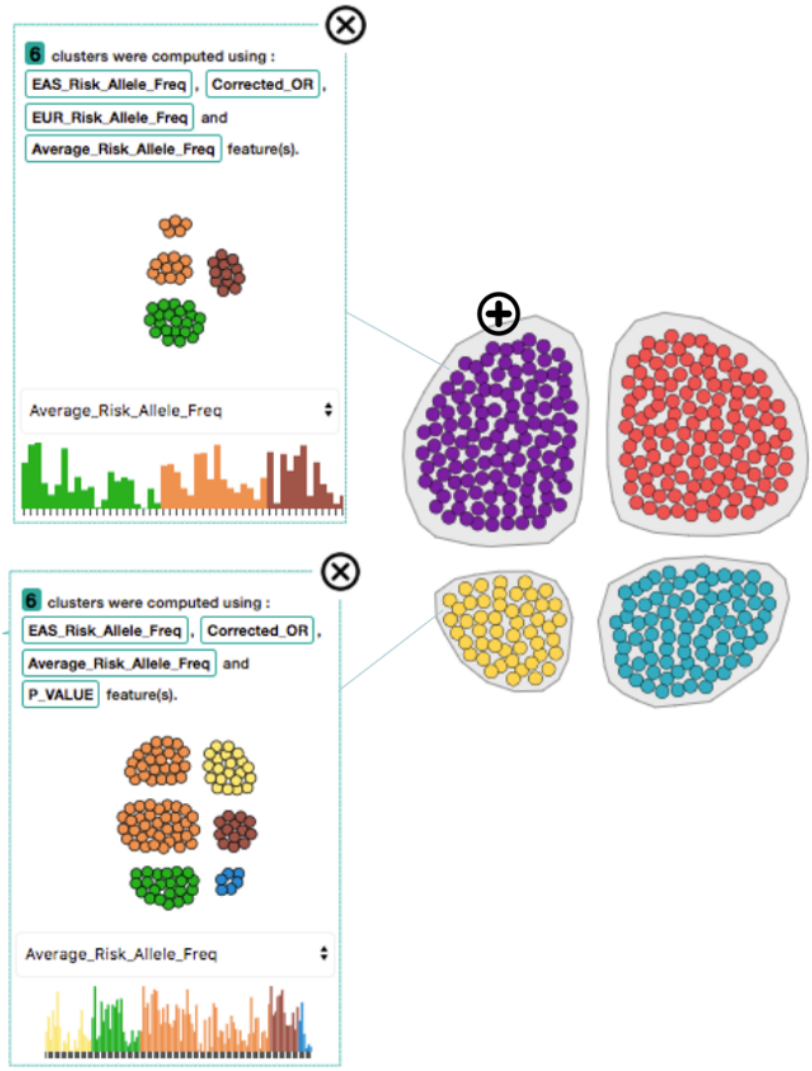


Figure 6.4: Megan clicks on the "+" icon to open the sub-cluster panel for clusters purple and yellow. Bar chart views showing comparisons of the feature Average-Risk-Allele between these clusters.

Africa with ancestory *ANC* has much higher disease risk factor than those sampled from other regions with ancestry *DER*.

Megan decides to merge the clusters *Blue* and *Purple*. To do so, she demonstrates her interest in merging the clusters by drag-drop the *Blue* cluster on the *Purple* cluster. In response, the system recommends new cluster layouts on the Recommendation Panel. She previews the thumbnails from the recommendation panel and selects the second recommendation, which results in placing 3 clusters in the Cluster View. To continue discussing the findings and implications with other colleagues, she exports a *.png* screenshot of the

current cluster layout. She also saves the results as a *.csv* file to investigate them more in other programs like R and SPSS.

6.1.5 Views and User Interface

Geono-Cluster's interface consists of: a Cluster View, a Recommendation Panel, a Table View, and an Attribute Panel. See Figure 6.1.

Cluster View visualizes the clustered data as Figure 6.1 shows. For testing their hypotheses, biologists often perform actions at the level of data items (e.g., move data items from one cluster to another). We visually present each cluster and its members on the Cluster View. The colored circles in each group represent members of a cluster; the surrounding hull represents the cluster. Users can hover over a circle, which prompts relevant attribute details of the data. Users can specify the number of clusters using the slider shown on the top-left. Cluster View is an environment similar to a spatial workspace in which users can move data items to structure their information and provide visual demonstrations **(G1)**. For example, a biologist might notice a set of data items should not be in a specific cluster. Thus, she can demonstrate that those points belong to a different cluster by dragging them from one cluster to another. The system uses the visual demonstrations provided by the users to steer the underlying recommendation engine **(G2)**.

Recommendation Panel shows different clustering results. Based on users' demonstrations on the Cluster View, the system recommends a set of appropriate clustering outputs. To compute the recommended clustering results, the underlying recommendation engine takes into account different (1) clustering techniques/algorithms; (2) combinations of attributes/features; and (3) clustering hyperparameters (i.e., varying 'k' for k-means clustering technique). Read section 6.1.7 for more details.

During the design process of Geono-Cluster, we examined different ways of presenting recommended clusters. We first considered showing all the recommended clustering results as small thumbnails in the Recommendation Panel. The biologists liked the idea and

the way that we recommended clustering results. However, the main challenge that biologists encountered was that they were not able to infer detailed information from the small thumbnails. Thus, they requested adding textual description of details about each clustering result in the recommendation. Currently, each thumbnail includes a textual description about the number of clusters, features used to compute the clustering recommendations, and a visualization of the clustering result **(G3)**.

Initially, we designed the recommendation module to update the view with new clustering recommendations whenever users show their demonstrations and/or adjust feature contributions. However, our users revealed that such approaches may distract their ongoing investigations on the current results. Thus, we compute cluster recommendations in the background but do not show the results immediately. Once the computation is done, a notification pops up, encouraging users to explore the results on demand by toggling the 'show recommendations' button (see Figure 6.1).

Table View shows a tabular representation of the loaded dataset where each row is a data item (see Figure 6.1). The initial version of Geono-Cluster did not include the Table View. However, biologists requested adding this view since it enabled them to check the raw data. The table updates on user's interaction on Cluster View. For example, selecting the hull of a specific cluster updates the Table View to show the details of items in the selected cluster. Users can click on a cell to find similar rows whose value are similar to the value of the item in that cell. This operation works on both quantitative and categorical data types. For categorical and ordinal data attributes we consider two data items similar if there is an exact match between them. For the numerical variables we define a threshold to measure similarity between two values. This technique allows users to filter and select a subset of data instances (enables selection of multiple rows simultaneously). From the design study we noted that the existing workflow of the biologists, involved exploring the data in MS Excel, then using "R" to run clustering models, and then export the data back to MS Excel for further analysis. We sought to provide a similar workflow under one system.

Attribute Panel lists the attributes of the loaded data set as Figure 6.1 shows. Users can turn on and off a set of attributes which directly affects the clustering algorithm. Furthermore, users can also adjust attribute contributions, specifying relative importance of the selected attributes to define cluster memberships **(G2)**.

6.1.6 Interactions

In this section we discuss how Geono-Cluster supports interactive operations commonly performed by biologists.

Merging and Splitting Clusters (T1): To merge two or more clusters, users first click on a cluster. They then demonstrate their interest in merging two clusters by drag-and-dropping the cluster on top of another cluster. Users can drag point(s) out of the cluster and drop into either i) another cluster or ii) a blank space (on the Cluster View). Drag-and-drop items into blank space is translated as forming a new cluster of the selected items outside the current cluster (see Figure 6.2). Demonstration-based cluster customization enables users to interact with the data directly and removes any mid-level instruments such as control panels or menus.

The merge interaction is derived from the previous work by Sarvghad et al. [14], in which they enabled HIV researchers to merge bars in bar charts by dragging one bar and dropping it over another bar. Biologists liked this interaction design and found it "direct and intuitive". To split clusters, we initially enabled biologists to select the data items by clicking on each circle representing a data item. However, biologists found it cumbersome and time-consuming. So, we implemented the lasso-selection such that users can select multiple data items easily. This operation allows user to brush over a set of data samples (represented as circles) in the Cluster View. In response the system extracts those samples from the current cluster and places them in a new cluster. If data samples from multiple clusters are selected (using lasso selection), then the system makes a new cluster from these lasso picked data samples.

Sub Clustering (T2): Hovering over a cluster reveals a plus button. Users can click on it to open a subcluster panel on the Cluster View, which shows subgroups of the data items within the selected cluster. In addition, a bar chart shows the distribution of a chosen attribute. Alongside, text description highlights the attributes that were used to compute the sub-clusters. Given that the users are not experts in data science, we do not present the quality metrics (e.g., silhouette scores, homogeneity score, etc.) Instead, we describe cluster models by showing thumbnail previews of clustering results with text descriptions as Figure 6.4 shows.

Delete data items or Clusters (T3): Our discussion with biologists revealed that they sometimes need to 'exclude' data items or clusters from their analysis while testing a hypothesis. Thus, we initially implemented the 'delete' feature by enabling users to select a subset of items or clusters from the main view and click on the delete icon. However, when we showed it to the biologists, they had trouble due to inconsistencies between the button-based interaction and other demonstration-based interaction.

Currently in Geono-Cluster users cam drag-drop a selected cluster on the delete icon shown on the top-left of the interface to show their interest in moving the selected cluster out of the layout. Similarly, they can drag-drop individual data items to demonstrate their interests in removing them from the cluster assignment.

Creating Customized Clusters: Users can select a subset of data items by clicking on the rows shown on the Table View (each row represents a data item). After selecting a subset of rows, users can drag-and-drop them on the Cluster View to demonstrate their interest in creating a clustering, in which all the selected data items fall in the same cluster. Users can iteratively repeat the process, and each drag-and-drop operation forms a new cluster in the Cluster View. Participants liked this idea as they found the design and the workflow of this interaction consistent with other interactions.

6.1.7 Computational Techniques

This section describes the underlying computational techniques which enable Geono-Cluster to recommend cluster models by incrementally steering (multiple cluster models) them to adhere to demonstrated user preferences. Our cluster model recommendation process includes the human in the loop. On a high level, the user shows their intentions on a cluster layout. Based on the operations, Geono-Cluster models multiple cluster algorithms and finds top k closest cluster models to the users' intention. Then, the user can refine the results through a series of customizations (instrumented through the interactions described above). In response, Geono-Cluster automatically finds close variants of cluster models and updates the recommendations in the Recommendation Panel. In summary, the system finds a set of cluster models with a distance function that reflects user-demonstrated cluster assignments.

Multiple clustering models: The clustering task begins when the user requests a new cluster layout (when they press the cluster button in the interface). In response, Geono-Cluster generates multiple clustering models M. Each cluster model M_i in M (M_1,M_2,M_3,M_4, ... M_T) is defined by a careful combination of a learning algorithm ω_i and a set of p hyperparameters ϕ, defined as ϕ_{i1}, ϕ_{i2}, ϕ_{i3}, ϕ_{i4}, ... ϕ_{ip}. Applied clustering algorithms include K-Means, DBScan, Agglomerative Clustering, and Spectral Clustering. In the evaluation of the system, we used K-Means cluster model as we observed through our design-study that most of our users are familiar with packages in "R" to use K-Means clustering (with default parameterization) to cluster the genome data. However, depending on the need of the user and the data used, Geono-Cluster can be extended to use other clustering methods. Nevertheless, each algorithm has its hyperparameters. For example, K-Means is a learning algorithm with "k" and the "max-iteration" value as an input hyperparameter. Furthermore, each model M_i in M is assigned a metric score S_i to compute S, which defines the quality of the clustering output (a higher S_i means a better cluster definition). Geono-Cluster

uses Scikit-Learn's ML package to construct and evaluate the cluster models using various quality metrics (e.g., Silhouette Coefficient, Davies-Bouldin index)

Recommendation Technique: Geono-Cluster ranks the models in M by their scores S explained below, and visualizes the best clustering layout in the Cluster View. Further, the system allows the user to inspect top f best cluster models from the ranked models M, through the Recommendation Panel (see Figure 6.1-a). If a user makes any customization to the shown cluster model M_c (e.g., merge or split clusters), the system automatically updates the recommendations by computing a new set of M cluster models, except the model M_c, which is currently shown in the Cluster View. Per iteration, the system updates S and the ranking of the models M based on user interactions with the data. Next it visualizes the best model in M in the Cluster View and shows thumbnail previews of the top f models in the Recommendation Panel (**G2**).

Geono-Cluster's model recommendation finds the closest fitting cluster assignments, whenever the user customizes the current cluster layout in the Cluster View. However, there can be scenarios that no cluster recommendation matches the user's intended changes. This may occur when users seek clustering results, which are mathematically infeasible. There could be various reasons for it, such as, users may have a different understanding of the data than what the data actually contains, or the data may have noise, etc. In such cases, users may need to be educated to understand the reasons for a different clustering result, which we plan to integrate in the workflow in the future. Currently, in such cases, Geono-Cluster still responds with the nearest best clustering output, though it may not resemble the layout shown by the user. Furthermore, to ensure users can see unexpected clustering results (to ideate and explore), every few iterations the system also recommends a set of cluster models that are randomly parameterized and thus clusters the data in an unexpected way. While our approach may seem similar to active learning (AL) [138] as in both, users specify feedback to the input data that drives the generation of a model. However, in our case the data does not have class labels. Furthermore, in AL, the system "asks" users to

give feedback on specific data points, while in our technique users have the freedom to interactively explore and provide feedback any time along the process.

Clustering Metric: Initially the system does not have cluster assignments or labels for any of the data instances. Thus to compute S the initial cluster models M, are evaluated using the Silhouette Score metric [139]. This metric is computed using the mean intra-cluster distance, and the mean nearest-cluster distance for every data instance. As users interact and assign clusters to a set of data instances I, Geono-Cluster applies two types of metrics to calculate S. To compute the first metric S_1, the system finds all the correlational features fc_k and non-correlational features $fc_k^{'}$ that describes each cluster ($k = 0$ to g clusters). Here, fc_k and $fc_k^{'}$ defines how the user characterises each cluster. Next, when M is computed the system computes the correlational features fc_{ik} and $fc_{ik}^{'}$. The system compares fc_k with fc_{ik} (and $fc_k^{'}$ with $fc_{ik}^{'}$) for each model in M to derive the clustering metric S_1, that describes how closely the cluster model M_i adheres to the clusters defined by the user (S_1 is normalized between $0 - 1$, higher is a better model). The second metric S_2 is based on the labels assigned to data items by users for I. The system finds other data instances $J = N - I$ (N is all data instances) that are similar to the user interacted data instances using cosine similarity distance metric based on their attribute values (categorical variables are one-hot encoded). The system automatically assigns to these similar data instances (J) the same class assignments that the users assigned to I. Next using this labelled data, the system applies Homogeneity index score [139] to compute S_2. This metric uses true labels and assigned labels by the system (when a cluster model M_i is applied to the data) to give a score to each model in M. The final score S is defined as the weighted linear combination: $S = \lambda_1 * S_1 + \lambda_2 * S_2$. Here the weights λ_1 and λ_2 are hyperparameters that are assigned based on how well the clustering outputs satisfied the biologists expectations.

User driven feature selection: A cluster model M_i is driven by a set of features F = f_{i1}, f_{i2},f_{i3},f_{i4} ..... f_{ik} as input to compute the distance function which assigns a set of

data items D to individual clusters C. In Geono-Cluster, the set of features F is either computed using feature selection methods e.g., "select K Best" [140], "PCA" [141] or can be retrieved from users if they specify a set of features and their relative weights (from the Attribute Panel supporting the task **T3**). When users specify a set of k features F_u = f_{i1}, f_{i2},f_{i3},f_{i4} ... f_{ik} with respective weights for each feature (W_u = w_{i1}, w_{i2},w_{i3},w_{i4} ... w_{ik}, the system updates the distance function in the clustering algorithm. The distance function is represented as $\sum_{i=1}^{k}\sum_{j=1}^{n}\left\|x_i^{(j)} * w_i^{(j)} - c_j\right\|^2$, where c_j, is the jth cluster centroid and $w_i^{(j)}$ is the user assigned feature weight.

Sub Clustering: When triggered by users, the system builds a sub-cluster model M_{si}, for data instances E, member of a selected cluster C_i. Unlike the set of main cluster models M, only a single sub-cluster model is generated per cluster (**T2**). For sub-clustering we relied on the parameterization of the best-recommended cluster model for the entire data i.e, best-found parameterization of the K-Means cluster model. To avoid further compute times that may impact real-time interactions, we did not construct and test multiple cluster models for sub-clustering. However, clicking on the "add subcluster" button again for the same selected cluster C_i, the system recomputes the sub-cluster model M_{si}, by randomly choosing a new set of a learning algorithm ω and hyperparameters ϕ; e.g., it picks a new "k" on the "K-Means" cluster model. This technique allows users to rapidly browse a large set of sub-cluster models.

Similar item selection: Users click on a cell (q_j) of a quantitative attribute on the Table View to select a value v_j of the data item d_i. Geono-Cluster finds a set of r data instances, U = d_a,d_b,d_c ... d_r, each of whose value v_j falls within a threshold range, say $[+eps, -eps]$. The parameter eps is set for each quantitative attribute Q by heuristics and can be adjusted. This technique allows users to pick data instances which are similar, based on the selected quantitative attribute q_j. Further, users can select another quantitative attribute cell q_k. Next, from the set of selected data instances U, the system finds all instances V which fall within a threshold range of the value selected for attribute q_k. Here the size of V is less

than that of U. This technique allow users to filter and select a subset of data instances V from the Table View. For categorical features X, Geono-Cluster performs exact feature value matching instead of matching data items based on a predefined range. Users can drag-drop these V data items to the Cluster View as a single cluster ($C = C_1$). They can continue selecting another set of data items, then add them to the cluster view as a new cluster ($C = C_1, C_2$). Users complete the data exploration or they can request the system to find a model M_i iteratively (**T3**).

Scalability: Unsupervised learning is expensive. As the number of data items increases, the cost of cluster assignments also grows higher. Geono-Cluster can run cluster computations for approximately 3000 data items without major delays. For the scope of our study, the number of data items seem practical.

6.1.8 Evaluation

To evaluate Geono-Cluster, we performed a qualitative assessment with six biologists to collect subjective feedback and observational data. Our study had two main goals: (1) collect qualitative feedback on Geono-Cluster's features and design, and (2) observe how experts perform visual clustering analysis using Geono-Cluster. In particular, our study indicates how Geono-Cluster helps domain experts gain insights into data by interactively building clusters.

Participants and Setting

We recruited 6 biologists (2 female, 4 male), all with graduate degrees related to Biology, Bio-Statistics or Bio-Informatics. They had $1 - 2$ years of experiences working with *Gene* related datasets. They had not participated in our preliminary evaluation of Geono-Cluster and were also not involved in the design of Geono-Cluster. All participants were familiar with the concept of data clustering and had previous experience with data grouping with at least one data analysis tool (e.g., SAS, R, etc.). Further, as they had previously worked

with *GWAS* catalogue data, they were familiar with all the data attributes in the dataset. During the entire study participants used a computer with 17-inch screen and used a mouse to interact with the system. The study took approximately 50 minutes and we rewarded each participant with a $ 20 gift card.

Procedure

Introduction and Training: Participants were briefed about the purpose of the study and their rights. After filling out the study consent form and a questionnaire on demographics, we asked participants to watch a tutorial video of Geono-Cluster. The video walked the participants through different features and interactions provided by the tool. After watching the video, we asked participants to work with the tool for 10 minutes. We encouraged the participants to ask as many questions as they want during this stage.

Main Study: The participants were asked to explore the *GWAS* Cataloge [128] data that includes published *SNPs* and association studies. In particular, we asked the participants to imagine their colleagues asked them to analyze the dataset using the visualization tool for 30 minutes and report their findings. Participants were instructed to verbalize analytical questions they have about the data, the tasks they perform to answer those questions, and their answers to those questions in a think-aloud manner. In addition, we instructed them to come up with data-driven findings rather than making preconceived assumptions about the data. The interviewer played a role of 'active listener' during the study. He facilitated participants' verbal reports by asking questions like "what are you trying to do?", "what are your thoughts now?", "what do you think about current groupings?". We tried to avoid interrupting the participants as much as possible during their data exploration process. However, we sometimes reminded that this is a think-aloud study and they need to verbalize their thoughts.

Follow-up Interview: After each participant completed the task, the experimenters asked participants to explain major obstacles of the tool and describe what they liked or disliked about the tool.

Data Collection and Method of Analysis

We screen and audio-recorded the whole study. During the main study, the experimenter took notes while participants interact with the system. We also collected feedback from a semi-structured interview with open-ended questions at the end of the study. We analyzed around 300 minutes of screen-capture videos from six participants. First, one of the authors transcribed the audio recording of the study. Then, two coders (first and second authors) read the transcribed data (including the think-aloud sessions and the interview responses) to parse a set of meaningful text snippets. After reading the data, each of the coders independently assigned codes (a word or phrase) to best describe the text snippets. Finally we consolidated the codes from the two authors by focusing on the aspects of the responses which highlighted positive or negative feedback with respect to *usability of the system*, *easy of use*, *learning curve*, *future feature requests* or *strategies pertaining to exploratory data analysis using clustering models*. In the following section, we use **P1** to **P6** to respectively denote the participants one to six who participated in the evaluation.

6.1.9 Results and Feedback

Overall, all participants found Geono-Cluster easy to use and effective in performing cluster analysis tasks. Below, we categorize and discuss the findings of our qualitative study in more details.

System usability: All participants found Geono-Cluster's workflow easy to use, intuitive, and engaging. P2 remarked *"I can keep trying new ideas to quickly test different ways to*

cluster this data." P4 said *"It's so easy to use, I can quickly iterate and learn about the data much faster, than using packages in R to cluster data."* Further, many other participants found visualization to be a very good medium to learn about the data by exploring different clustering results. P5 said *"I never knew that I can use visual methods to explore clustering result. Currently I use R to cluster my data, then export a CSV file to my teammates."*

Consistency with user mental model: Participants found the design and workflow of Geono-Cluster consistent with their mental model and expectations. In particular, participants found that it is intuitive to visually demonstrate tasks such as creating, merging, and splitting clusters by demonstration. For example, P3 mentioned: *"it feels intuitive to merge clusters by dragging and dropping one cluster over another one. [...] this is what I would expect to happen."* P5 stated: *"I liked the idea of creating a cluster of items by moving the data items from this table to the empty space [dragging the data items from the Table View and dropping them on the Cluster View to create a cluster]."* Further P2 added: *"Compared to programming, using this kind of tool is more straight forward and faster."* Consistency and natural mapping between user's intent and the actions required for performing the intent is important in designing new interactions.

Perceived control over data analysis process: While using Geono-Cluster, P1, P4, and P5 commented on their level of control over the data analysis that resulted from their freedom in interacting with visualizations instead of going through layers of menu items. For example, P1 mentioned: *"This is great because I can construct my own cluster and tell the system how I want my clustering outcome looks like."* P4 stated: *"It is a powerful idea to enable analysts to use their knowledge about the data items to interactively create clusters [visually demonstrate their expected clustering outcome]. I specifically like how this allows merging and splitting clusters."* The level of interaction directness [87] with the vi-

sual representation contributes towards increasing the perceived control of the participants over the data analysis process.

Difficulty in splitting a cluster: Participants found the lasso interaction intuitive and easy to use. However, with lasso selection participants were not very exact about the data items that they wanted to select. For example, after selecting a subset of data items, P3 noted: *"It is hard to be exact with this selection. I don't want this specific point to be selected."* In such cases, participants had to either deselect the items that were selected incorrectly by clicking on them or try to lasso select again. Going forward, we envision designing advanced interaction techniques for easier selection of data items that are located in a close distance from one another.

Interpretability of recommendations: Although some participants liked how the recommendations were presented, two participants could not immediately understand why specific recommendations are suggested. For example, P2 mentioned: *"I understand what each cluster represents which is good, but I am not sure why these recommendations."* and P3 stated: *"I am curious how these recommendations are added."*. Going forward, we suggest systems to explore design alternatives to explain the reasoning behind recommendations. In situations when the system does not find any cluster recommendations that matches user's demonstrated changes, Geono-Cluster shows the nearest best clustering layout. In such scenarios, users may be surprised to see the abrupt or strikingly different recommendations. In the future, we are thinking of explicitly communicating this conflict in textual description. At the same time, we want to introduce a more variety of models so that the system can perform deeper search to find desirable results.

Observations

Our user study reveals that participants usually began exploring the data by framing a hypothesis, asking the questions they want to know, and then performing a set of tasks (as described in section 6.1.1) through Geono-Cluster's interface to find the answers. Interestingly, we observed that participants often took two different approaches to perform visual data clustering: **Top-down** and **Bottom-up**. Below, we describe each approach in more details.

6.1.10 Top-down Visual Data Clustering Approach

P1 started his data analysis process by asking *"How does the gene samples differ in disease risk factor by regions and chromosome factors?"* To that end, P1 clustered data items by selecting a set of features from the Attribute Panel and then pressed the *Cluster* button. Next, he checked the recommended cluster layouts from the Recommendation Panel to explore other clustering results based on another set of features. In response, he updated the list of features to cluster the data by and triggered Geono-Cluster to generate a new cluster layout. P4 also followed the same approach; however, he did not have any question to begin with. He initialized the process by pressing the cluster button to start with an initial clustering. Next, he hovered over data items in each cluster to familiarize himself with the data items and find similarity or dissimilarity. He also checked the Table View to compare different data items from various clusters. If the clusters did not match his mental model, he would adjust the features from the Attribute panel. He would then preview the recommended clustering options to further explore a wide range of cluster outputs. This process continued until he was satisfied with the clusters and had a better sense of the data.

A main point here is that in the top-down approach participants mostly avoided interaction at the data item level, but instead they dealt with the full range of features from the Attribute Panel. P1 also verified this point by saying: *"I relied on cluster button to cluster the data, as I do not specifically know much about the data items, so did not use the table's*

drag-drop feature. Similarly, I did not customize the clusters by using lasso or drag-drop feature initially. I rather re-computed the clusters based on a new set of features that I specify." However, P1 later confirmed that over iterations when he was more confident about the data, he started using the split and merge operations to customize shown clusters.

6.1.11 Bottom-up Visual Data Clustering Approach

Remaining participants (P2, P3, P5, and P6) followed the Bottom-up approach, in which they mainly relied on interaction at the data item level. They first created a customized cluster by dragging data items from the Table View and dropping them on the Main view as opposed to relying on the cluster button. These participants often interacted with data items to demonstrate their expected outcome.

P2 started her clustering analaysis by asking *"How does the gene samples derived from humans/monkeys (ANC) vary from gene samples derived from mixing humans and monkeys (DER) with respect to various diseases?"* To answer the question, P2 placed all the *ANC* gene samples into one cluster and a few *DER* gene samples into another cluster from the Table View. P2 remarked: *"my strategy is to select a set of data points [items] based on the gene's ancestry, then drag-drop to create a cluster"*. P2 then previewed the recommendations to explore other options to cluster the data based on his specification of clusters. In this process, P2 did merge/split clusters to test different ideas to cluster the data using the lasso-selection and the cluster drag-drop feature. P2 said: *"I also rely on the lasso tool to define other clusters from this, if the cluster appears too big"*. Using Geono-Cluster, many participants were able to customise clusters in this fashion to find interesting insights from the data, that they found needed further analytical investigations/research with their peers or mentors. For example, one participant was able to find a significant difference in *Average-risk-allele-frequency* between two sets of clusters by iteratively following this bottom-up visual clustering approach.

P6 also followed the same approach. P6: *"I want to know if the gene with chromosome*

factor higher than 6 sampled from America, have higher cancer risk factor? To seek an answer, I find the Table View's data item selection feature quite useful, as I can define my own clusters based on chromosome value or the region the gene was sampled from." We noticed that when P6 explored the initial set of cluster layouts, he paid attention to the suggested features (in the Recommendation Panel) to understand how the cluster is defined. In some cases, P6 did not agree with the recommendations or the features that were used to derive the results. To provide his feedback for updated results, he customized the best-perceived cluster layout by splitting the existing clusters using the lasso tool and merging smaller clusters into one. P6 added: *I am using the lasso feature to take out all the data items which have chromosome value less than 6. Also, the smaller clusters with 5 or fewer data samples are confusing, so I merge them into one*.

6.1.12 Discussion

In this section, we discuss lessons learned from our study. We also discuss limitations of our approach for future studies.

Generalizability of the approach: The methodology for this work stems from a design study [127] in biology. This inherently makes our contribution domain-specific, and solves a very specific problem that we discovered through working with biologists. However, the underlying interaction technique behind Geono-Cluster is generalizable and can be applied in other domains and on other tabular datasets. Our demonstration-based interaction design is a bottom-up approach where users provide demonstrations by interacting with data items. As such, this technique works whenever users can bring their knowledge by interacting with data points and provide demonstrations. For instance, another dataset that biologists use is cancer dataset to cluster patients based on the likelihood to be diagnosed with cancer. In this case, the domain experts may know patients with certain chromosome value or blood count level.

Human bias in interactive clustering: The human-in-the-loop nature of Geono-Cluster

introduces potential user biases in visual data exploration. In fact, some amount of human bias exists in most interactive systems (e.g., control panel style interfaces). However, the key goal of Geono-Cluster is to help users explore different aspects of data while testing their hypotheses using clustering models. More importantly, the goal of our tool is not to help users build the most accurate cluster model, but rather to: (1) explore alternate models and their outputs, and (2) validate these models based on metrics that are meaningful to them (instead of relying on conventional metrics). The results of our user studies also show that biologists using Geono-Cluster successfully gleaned insights from the data and learned about their data at the end of their exploration process, and not just construct a set of clustering models.

Extending current interactive clustering approaches: Previous interactive clustering tools (e.g., Clusterophile [142] etc.) follow a top-down approach, where users define cluster parameters through control panels to build cluster models. To interact with these tools, users should know various cluster parameters and how to adjust them. Instead, GeonoCluster follows a bottom-up approach in which it enables users to apply their domain knowledge by interacting at the data instance level (without having to learn model parameters or metrics), and the system infers users' intent from the given demonstrations to recommend cluster models. Also, unlike other interactive clustering tools, our work solves a specific problem in a domain that we discovered through working with biologists.

Model Feedback and Interpretation: Periodic discussion and informal inputs from the biologists clarified that model interpretation and feedback (to the model) is of critical value to them. For example, when Geono-Cluster shows a set of clustering recommendations, users may need to know how they differ from each other, or what logic was implanted to define the displayed clusters. There are many ways to explain this to the user; however, we only selected methods which do not require any technical expertise from the user. Our final design explains a cluster by using a natural language-based approach to communicate the features that were used to compute the clustering distance function. In particular, we

avoid showing technical information such as silhouette coefficient or exact feature weights to provide a high-level model explanation that does not overwhelm the users with a bag of information that might not be easy to interpret. Our qualitative feedback hints that our approach made Geono-Cluster not only easy to use but also an engaging tool to continue data exploration by rapidly testing different ideas to cluster the data.

Cluster Model Comparison: While representing multiple clustering results show different ways to partition the data, model comparison to understand trade-offs between these clustering options is critical. However, in our current prototype we do not support explicit cluster model comparison. For example, users cannot perform a pairwise comparison of two cluster models side by side [143], or they cannot select a few chosen cluster models to see the results in a way which facilitates direct comparison. Based on our interviews with the biologists, comparing cluster models was not posed as a requirement to us. Therefore, we deliberately did not include cluster model comparison as one of the design goals of the system. However, as visual analytics researchers, we understand that being able to compare multiple cluster models, may positively aid model selection and enhance the tools use case.

Limited Model Explanation: Geono-Cluster explains a cluster model by highlighting the top k features that were used to compute the underlying distance function using a natural language expression. Though the simplicity of the explanation is helpful for non-experts, in certain cases this may pose as a very limited explanation of a clustering model. For example, two cluster models may be based on the same set of features, but the defined clusters are strikingly different. In this case, users may get confused to interpret the difference between these models. We reckon this as a limitation of our tool which we intend to study in future work.

Scalability: The current interface and the supported interactions (i.e., split and merge technique) is tested with 3000 (approximately) data items. However, we understand that as the size of the data grows, the interaction techniques such as drag-and-drop interaction and lasso-selection tool may be less responsive. In the user study, P6 noted that the lasso-

selection was less effective for large clusters when the data items became too small to select or notice (often partially obscured by neighboring data items). We envision multiple ways to enhance scalability in such demonstration-based systems. For instance, we could aggregate data items as defined by the user, e.g., group all data items with *Chromosome* value greater than 7.

CHAPTER 7
DISCUSSION AND CONCLUSION

This thesis contributes a novel interaction paradigm for data analysis called **"Visualization by Demonstration (VbD)"**. In my thesis, I first investigated the principles of VbD [144, 65, 66], developed and evaluated general-purpose visualization systems to investigate opportunities and challenges in systems that implement VbD [145, 146, 147], and finally applied VbD to specific domains such as biology [8].

In a series of studies we developed fundamental principles and guidelines that go into design of VbD. For example, I conducted an experiment to understand how fast and accurate participants can manipulate 12 different interactive visual marks (e.g., length) [65]. Results of this study showed that participants had above 80% accuracy in adjusting a majority of visual marks. In a different study, I conducted an elicitation study to investigate what type of direct manipulation strategies participants employ to visually demonstrate their intended changes [66]. Based on findings of this study, I identified a list of visual demonstrations people employ to perform each operation and derived implications to help designers leverage direct manipulation of visual mark for providing demonstrations.

To examine the feasibility of VbD, I designed and developed *VisExemplar* [145]. VisExemplar is a general-purpose tabular data visualization tool that implements VbD. It allows users to create visualizations and perform different visualization tasks using the VbD paradigm. VisExemplar allows users to provide visual demonstrations by directly manipulating the visual representation. It then infers users' intention and generates possible visualization mappings in response to the given demonstrations.

To empirically evaluate VisExemplar, I conducted a series of comparative studies [146] using two tools: one implementing the commonly used Manual View Specification paradigm (Polestar) and another implementing VbD (VisExemplar). My goal was to investigate the

trade-offs between these two paradigms. These studies showed that both paradigms have their own advantages and disadvantages, and suit different types of tasks. Manual View Specification tools are fast and have high external consistency. On the other hand, the free-from interaction in VbD systems leads users to have higher interaction expressivity and be more creative during data analysis process. Trade-offs between the two paradigms motivated me to investigate how complementary these two paradigms are, and if it is possible to leverage the benefits of both paradigms by combining them into a unified tool.

To study opportunities and challenges in combining two interaction paradigms, I designed and developed *Liger* [147], a multi-paradigm system that combines Manual View Specification and VbD in a unified tool. Through the design and implementation of Liger, I investigated how interaction paradigms can be blended to generate context that complements the individual paradigms. I then conducted a qualitative study of Liger with 10 participants, providing initial evidence that people 1) use both paradigms interchangeably, 2) seamlessly switch between paradigms based on the operation at hand, and 3) choose to successfully complete a single operation using a combination of both paradigms.

Finally, I collaborated with biologists at Georgia Tech and health informatics researchers at IBM Research to design and build Geono-Cluster, a novel visual analysis tool designed to support cluster analysis for biologists who do not have formal data science training [8]. Geono-Cluster enables biologists to apply their domain expertise into clustering results by visually demonstrating how their expected clustering outputs should look like with a small sample of data instances. By translating these demonstrations into target similarity functions, the system predicts users' intentions and generates potential clustering results. Using the system, users can create, evaluate, and refine clustering results in order to gradually reach the most optimal clustering result for the users' analysis goals. Geono-Cluster is currently deployed and used by biologists at Georgia Tech.

In summary, the contributions of this thesis are:

- A new demonstrational interaction paradigm (Visualization by Demonstration) that

enables people to visually explore their data

- Empirical evidences describing the trade-offs of Visualization by Demonstration
- Empirical evidences describing the effectiveness of direct manipulation of graphical encodings as method for providing demonstrations.
- A prototype system that enable biologists to perform interactive visual data clustering using Visualization by Demonstration

Chapter	Research Question	Publication
Chapter 2	**RQ1:** What are the fundamentals of the visualization by demonstration paradigm?	**Saket** et al. Demonstrational Interaction for Data Visualization, **CG&A 2019**
Chapter 4	**RQ2:** How do people demonstrate their intended goals using direct manipulation of graphical encodings?	**Saket** et al. Investigating Direct Manipulation of Graphical Encodings as a Method for User Interactions, **InfoVis 2019**
Chapter 4	**RQ3:** How effective is manipulating graphical encodings used in visualizations as a method for providing visual demonstrations?	**Saket** et al. Evaluating Interactive Graphical Encodings for Data Visualizations, **TVCG 2017**
Chapter 5	**RQ4:** How can we apply visualization by demonstration to design tools for data exploration?	**Saket** et al. Visualization by Demonstration: An Interaction Paradigm for Visual Data Analysis, **InfoVis 2016**
Chapter 5	**RQ5:** How effective is visualization by demonstration compare to traditional manual view specification?	**Saket** et al. Investigating the Manual View Specification and Visualization by Demonstration Paradigms for Visualization Construction, **EuroVis 2019**
Chapter 5	**RQ6:** How to offer the benefits of both visualization by demonstration and manual view specification in a unified visualization tool?	**Saket et al**. Liger: Combining Interaction Paradigms for Visual Analysis, **Under Review**
Chapter 6	**RQ7:** How can visualization by demonstration enable domain experts to perform real-world tasks such as data clustering?	**Saket*** and Das* . Geono-Cluster: Interactive Visual Cluster Analysis for Biologists, **Under Revision** [* **Equal Contribution**]

Figure 7.1: List of research questions investigated in this thesis.

7.1 Challenges in Interpreting the Provided Demonstration

VbD systems infer users' intentions based on the given demonstrations. This can be extremely challenging to infer users' intentions from such low-level data sources. Indeed, in some cases systems might not be able to interpret the provided demonstration. This raises a question: *What are the possible ways for VbD systems to react when they are not able to interpret a given demonstration?* While it is nontrivial to overcome this challenge completely, there are ways to help users encountering this challenge in VbD systems.

"Demonstration Unknown": One approach to this challenge is having the system simply responding with "demonstration unknown". This is very similar to how some of the programming by demonstration tools such as Peridot [43] handle cases where the intent functions are incapable of extracting the user's intention from the given demonstration. The system could require users to provide more demonstrations or different types of demonstrations when it is less confident about the interpretation of a specific demonstration.

Demonstration Recording Feature: Another way to address this challenge is to provide features that enable users to record their demonstrations and define new meanings for the given demonstrations. For instance, the user could first drag the tallest bar in a bar chart to the extreme left (interface could provide a button that enables the user to record her demonstration). She could then specify that the recorded demonstration should be mapped to sorting the bar chart. In the programming by demonstration literature, tools such as Tinker [43] support this capability by enabling users to define new demonstrations and map them to the expected meanings. Tinker also allows users to define a mapping between a demonstration that already exists and a new mapping. As such, the more users work with the system, the better the system gets in interpreting users' intents.

Multiparadigm Data Visualization Tools: Finally, one approach to address this challenge is to incorporate both VbD and MVS paradigms into one system. As such, users can utilize MVS to perform their tasks in cases where the system is incapable of interpreting the given demonstrations (assuming that MVS supports the given task). For example, Liger [147] is a multi-paradigm system that combines MVS and VbD in a unified tool. During the data analysis process, users can switch to MVS paradigm whenever the system is not capable to interpret their given demonstrations.

7.2 Demonstrations that Imply Multiple Meanings

A given demonstration could imply multiple meanings. For example, coloring a data point could mean user's interest in coloring all data points, coloring data points similar to a spe-

cific data point, mapping a data attribute to color encoding, etc. One of the main challenges in designing VbD systems is to decide how to handle cases in which a given demonstration implies multiple meanings. There are several ways to overcome this challenge.

Recommending Potential Options: One way to handle cases in which given demonstration implies multiple meanings is to recommend all/subset of potential meanings in response to the given demonstration. For example, VisExemplar [6] applies this method by computing and recommending multiple transformations in response to the demonstrations that imply multiple meanings. While recommending a set of potential meanings can enable users to explore a wide range of possible next steps, it requires users' time and effort to explore different options ans could results in a slower data exploration process [146].

One-to-One Mapping: Another solution to this might be to design the VbD systems in a way that there is only one interpretation for a given demonstration (one to one mapping). In such cases, upon providing a demonstration, the system extracts the expected meaning and computes the expected changes. For example, in Geono-Cluster [8] each unique demonstration is mapped to a task. For example, dragging a cluster and dropping it on another cluster is mapped to user's intention in merging the clusters. However, in reality, as the functionality of the visualization system increases, the idea of defining a unique demonstration for every task is becoming less practical (i.e. we cannot define a demonstration for every single task). Moreover, this increases the complexity of the VbD systems since users need to take a new set of actions to demonstrate different goals. This idea is probably more practical for the systems that are designed to support a limited set of task(s) for a certain group of target users.

Ask for More Demonstrations: The system could require users to provide more demonstrations when it is less confident about the interpretation of a specific demonstration. Early on during the process of providing demonstrations, mapping the given demonstrations to possible meanings might be more ambiguous, but as users continue completing their demonstrations and providing more evidence and training data to the system, the number

of possible transformations would decrease (and ideally become more accurate with respect to the users intents). For example, coloring one data point red could demonstrate user interest in coloring all data points red, assigning a data attribute to color, etc. However, coloring one data point red and another one blue can help the system to rule out some of the potential interpretations such as user's interest in coloring all data points red.

7.3 Multimodal VbD Systems

In this thesis, I mainly concentrated on investigating desktop-based VbD systems that use mouse and keyboard as the main input modality. Going forward, I want to investigate how we can incorporate other interaction modalities such as touch and pen to VbD systems.

During a visual demonstration, multiple valid interpretations of a user's action can be made. This ambiguity challenge for demonstration-based systems has been previously studied [16], and solutions for disambiguiation exist. While most systems use a fixed model for determining the most "appropriate inference", there is no guaranteed way to either identify the user's intent correctly or be able to resolve the ambiguity without further assistance [16]. In the case of VbD systems presented in this thesis, part of the ambiguity stems from the limited amount of information that direct manipulation of graphical encodings can convey. Going forward, the use of simultaneous modalities (pen, touch, speech, etc.) could decrease ambiguity and enhance users' experessivity by increasing the amount of information users can provide about their demonstrations.

Incorporating multiple input modalities can also increase a set of available intuitive demonstrations for users. There exists tasks that is intuitive to demonstrate them using mouse input (e.g., dragging the tallest bar in a bar chart to the extreme left or right of an axis to demonstrate their interest in sorting the bar chart [11, 12, 66]). However, there exist tasks that would be more difficult and less intuitive to demonstrate them using the mouse input. For example, we recently found that the users had difficulties in finding an intuitive visual analogy to demonstrate their interest in switching from a scatterplot visualization to

a bar chart using the mouse input [13, 66]. While it is hard to perform such operations using the mouse input, it could be much easier to demonstrate them using other modalities such as pen (e.g., users could draw a bar to demonstrate their interests in switching to a bar chart). Thus, depending on which tasks and operations a user wants to demonstrate can help determine which of the input modalities to use for the specific task.

Despite the benefits of incorporating multiple input modalities, there are challenges in such multimodal VbD systems. Multimodal VbD systems might require users to put more effort into thinking about the data and the task at hand, and selecting an input modality to demonstrate that task. This is likely to make the process of visual data exploration *slower* in such tools. However, a visualization is not just a means to an end. Reflection on the data, tasks at hand, and possible interactions also take place during data exploration [107]. Optimizing for efficiency may not lead to the overall best outcome, as it may result in users glossing over important details of the data, the operations, and how to best demonstrate the operations. Indeed, "slow" data exploration can results in users' active involvement, foster creativity and critical thinking, and encourage conscious, deliberate, analytical reasoning [108, 109]. We hypothesize that multimodal VbD systems can bolster deliberate and more logical processes by requiring users to think carefully about their tasks at hand and the different ways of achieving these tasks.

Switching between input modalities likely increases *interaction cost* [110]. When using an input modality over a certain period of time, people form habits and learn the design principles that drive this modality [111]. After forming a habit with one modality, switching to another modality requires breaking this habit and spending some time to recall the principles of the new modality. In addition, combining multiple input modalities can increase the number of functionality supported by a tool. We know that people should discover and learn the functionality supported by each modality as they use the tool over time [148]. However, discoverability and learnability of interaction still is an important challenge to address in visualization [99, 113]. Researching ways of improving *discoverability* and

learnability of multimodal VbD systems is a promising avenue for future work.

7.4 Power of the VbD Paradigm in Task Specific Visualization Tools

Based on my experience designing different VbD systems over the past few years, I believe this paradigm is most effective when applied to data analysis tools that support a small set of tasks. Designing VbD systems that support a small set of tasks benefits both developers as well as end-users. As I showed in tools such as Geono-Cluster [8], it is feasible to address challenges such as *lack of discoverability of possible demonstrations* and *incapability in inerpretability the given demonstrations* in VbD systems that support a small set of tasks. However, such challenges become much more difficult to address as the number of operations/tasks supported by the tool increase.

One could argue in which cases we need data analysis tools that support a small set of tasks. During my Ph.D., I have closely collaborated with experts in specific domains such as healthcare and biology and applied VbD to address challenges these experts regularly face during data analyses (e.g., [8, 14]. While working with domain experts in different disciplines, I noticed that there are often a subset of particular data analysis tasks that experts perform frequently (e.g., clustering [8], adjusting data groupings [14]). However, domain experts often encounter several challenges while performing their tasks. First, some experts do not have formal data science training. As such, they often use general-purpose data analysis tools that are designed for general audience such as SAS or Avantgarde. Such tools are often ill-equipped to help experts to perform domain-centered and advanced data analysis tasks (e.g., visual clustering). Second, experts with data science training often use programming languages such as Python and R to perform a variety of visual data analysis tasks. These languages often increase execution costs and impede visual data analysis for domain experts. Early on during the design process for building an interactive system for visual clustering [8], one of the biologists mentioned: *"my process [data analysis process] is sometimes slow. [...] I search for code snippets online. After finding the code, it takes*

2 or 3 trials to get the code working." Thus, a significant opportunity exists to design and develop VbD systems that support specific tasks performed by domain experts.

7.5 Expanding Visual Data Exploration in K-12 Settings using VbD

We encounter data visualizations online, in newspapers, or on TV on a daily basis. Visualization is a powerful medium to unpack complex data into simple insights and communicate them to a large set of audience — for instance, communicating presidential election results or data-driven insights to support positive health and enable self-awareness. It is therefore important for *everyone* ranging from high school students to analysts in large organizations to make sense of their data.

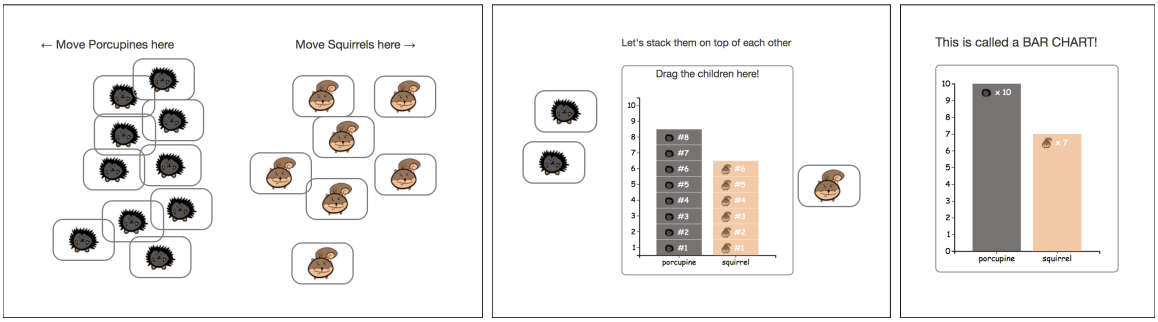
Unfortunately, much of the work in data visualization often targets a population of users who have some levels of visualization expertise and training. We also have a large set of visualization authoring tools that enable designers to construct customized data visualizations. However, to the best of my knowledge not much work has focused on designing digital visual data exploration tools for visualization novices, particularly children. The use of data for decision-making in educational institutions is neither a new topic nor an unknown practice. Today, building data analysis skills is part of modern school curricula. Schools often engage students in visual data exploration and analysis using their own real data. However, many of the existing data visualization tools are often ill-equipped to help students to perform data analysis tasks. So, how can we engage students in more free-form visual data exploration that are driven by their own mapping ideas?

In a class project, we designed a desktop-based prototype called MagicVis[1] to help elementary school students (children between 7 and 11 old) learn about basic visualizations like bar chart. We used MagicVis as a testbed to understand how we can enhance visualization literacy in students using a tool that implements VbD. MagicVis enables users to perform small tasks such as sorting data points. In addition, it enables users to create

[1] `https://magicvis.now.sh/story`

bar charts by stacking the data points. This involves using interactions such as a drag and drop and snap to the bar. The main task - stacking the data points - uses transitions to help users understand the relationship between data points and the visual representation. See Figure 7.2 for more details.

We also conducted a preliminary study with an elementary student (8 years old) to observe his experience with MagicVis. Our participant was able to learn about bar chart in an engaging and playful way using MagicVis. We also noticed that our participant preferred touch interaction over mouse input. A few times he moved the data points by touching the computer screen and trying to drag the point. Implementing touch interfaces to take advantages of a highly intuitive platforms, i.e. the touch interface, so that our tool is easily used by young users would be a large boon and a high priority for the project as we move forward.



(a) (b) (c)

Figure 7.2: a): The system randomly positions a set of visual glyphs representing porcupines and squirrels on the screen and asks the users to order them. Users should move the visual glyphs representing porcupines to one side and the ones representing squirrels to another side. **b):** The system then asks users to stack the visual glyph on the top of each other. **c):** The system teaches the users that the final visual representation they created is called a "bar chart".

This class project was just a small first step towards exploring visual data analysis in K-12 settings using VbD. I hypothesize the high level of flexibility and expressivity in VbD tools could make visual data exploration more accessible to students. Therefore, a significant opportunity exists to conduct a series of systematic participatory studies that are centered upon students at different levels (elementary, middle, and high school) to better

understand both social and technological constraints on the visualization design space for them. Different VbD-based solutions can then be designed and developed to facilitate the process of creating and reading visualizations for students.

REFERENCES

[1] C. Ware, *Information Visualization: Perception for Design*. Elsevier, 2012.

[2] S. Few, *Now You See It: Simple Visualization Techniques for Quantitative Analysis*. Analytics Press, 2009.

[3] J. S. Yi, Y. ah Kang, J. T. Stasko, and J. A. Jacko, "Toward a Deeper Understanding of the Role of Interaction in Information Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1224–1231, 2007.

[4] E. Segel and J. Heer, "Narrative Visualization: Telling Stories with Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1139–1148, 2010.

[5] J. W. Tukey, "Exploratory Data Analysis," 1977.

[6] B. Saket, H. Kim, E. T. Brown, and A. Endert, "Visualization by demonstration: An interaction paradigm for visual data exploration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 331–340, Jan. 2017.

[7] B. Lee, R. H. Kazi, and G. Smith, "SketchStory: Telling More Engaging Stories with Data Through Freeform Sketching," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2416–2425, Dec. 2013.

[8] B. Saket, S. Das, B. C. Kwon, and A. Endert, *Geono-cluster: Interactive visual cluster analysis for biologists*, arXiv preprint 1911.00988, 2019.

[9] H. Kim, J. Choo, H. Park, and A. Endert, "InterAxis: Steering Scatterplot Axes via Observation-level Interaction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 131–140, 2016.

[10] E. Wall, S. Das, R. Chawla, B. Kalidindi, E. T. Brown, and A. Endert, "Podium: Ranking Data Using Mixed-Initiative Visual Analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 288–297, 2018.

[11] D. Maulsby and I. H. Witten, "Watch What I Do: Programming by Demonstration," in, A. Cypher, D. C. Halbert, D. Kurlander, H. Lieberman, D. Maulsby, B. A. Myers, and A. Turransky, Eds., Cambridge, MA, USA: MIT Press, 1993, ch. Metamouse: An Instructible Agent for Programming by Demonstration, pp. 155–181, ISBN: 0-262-03213-9.

[12] R. Sadana, M. Agnihotri, and J. Stasko, "Touching data: A discoverability-based evaluation of a visualization interface for tablet computers," *arXiv preprint arXiv:1806.06084*, 2018.

[13] B. Saket and A. Endert, "Evaluation of Visualization by Demonstration and Manual View Specification," *arXiv preprint arXiv:1805.02711*, 2018.

[14] A. Sarvghad, B. Saket, A. Endert, and N. Weibel, "Embedded Merge Split: Visual Adjustment of Data Grouping," *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[15] R. Amar, J. Eagan, and J. Stasko, "Low-level Components of Analytic Activity in Information Visualization," in *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, IEEE, 2005, pp. 111–117.

[16] B. A. Myers, "Demonstrational Interfaces: A Step Beyond Direct Manipulation," *Computer*, vol. 25, no. 8, pp. 61–73, 1992.

[17] M. Brehmer and T. Munzner, "A multi-level typology of abstract visualization tasks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2376–2385, 2013.

[18] J. Preece, Y. Rogers, and H. Sharp, *Beyond Interaction Design: Beyond Human-Computer Interaction*. New York, NY, USA: John Wiley & Sons, Inc., 2001, ISBN: 0471402494.

[19] D. Saffer, *Designing for Interaction: Creating Innovative Applications and Devices*, 2nd. Thousand Oaks, CA, USA: New Riders Publishing, 2009, ISBN: 0321643399, 9780321643391.

[20] B. Shneiderman, C. Plaisant, M. Cohen, and S. Jacobs, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 5th. USA: Addison-Wesley Publishing Company, 2009, ISBN: 9780321537355.

[21] M. G. Helander, T. K. Landauer, and P. V. Prabhu, Eds., *Handbook of Human-Computer Interaction*, 2nd. New York, NY, USA: Elsevier Science Inc., 1997, ISBN: 0444818626.

[22] A. Sears and J. A. Jacko, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications (Human Factors and Ergonomics Series)*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 2007, ISBN: 0805858709.

[23] N. Elmqvist, A. V. Moere, H.-C. Jetter, D. Cernea, H. Reiterer, and T. Jankun-Kelly, "Fluid Interaction for Information Visualization," *Information Visualization*, vol. 10, no. 4, pp. 327–340, 2011.

[24] R. A. Becker, W. S. Cleveland, and A. R. Wilks, "Dynamic Graphics for Data Analysis," *Statistical Science*, pp. 355–383, 1987.

[25] A. Dix, J. Finlay, G. Abowd, and R. Beale, *Human-Computer Interaction*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997, ISBN: 0-13-437211-5.

[26] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice (2Nd Ed.)* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990, ISBN: 0-201-12110-7.

[27] (2018). Tableau Software, http://www.tableau.com/, (visited on 02/12/2016).

[28] C. Ahlberg, "Spotfire: An Information Exploration Environment," *ACM SIGMOD Record*, vol. 25, no. 4, pp. 25–29, 1996.

[29] A. Satyanarayan and J. Heer, "Lyra: An Interactive Visualization Design Environment," in *Computer Graphics Forum*, Wiley Online Library, vol. 33, 2014, pp. 351–360.

[30] D. Ren, H. Tobias, and X. Yuan, "IVisDesigner: Expressive Interactive Design of Information Visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2092–2101, 2014.

[31] G. G. Méndez, M. A. Nacenta, and S. Vandenheste, "iVoLVER: Interactive Visual Language for Visualization Extraction and Reconstruction," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI '16, San Jose, California, USA: ACM, 2016, pp. 4073–4085, ISBN: 978-1-4503-3362-7.

[32] N. W. Kim, E. Schweickart, Z. Liu, M. Dontcheva, W. Li, J. Popovic, and H. Pfister, "Data-Driven Guides: Supporting Expressive Design for Information Graphics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 491–500, 2017.

[33] Z. Liu, J. Thompson, A. Wilson, M. Dontcheva, J. Delorey, S. Grigg, B. Kerr, and J. Stasko, "Data Illustrator: Augmenting Vector Design Tools with Lazy Data Binding for Expressive Visualization Authoring," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18, Montreal QC, Canada: ACM, 2018, 123:1–123:13, ISBN: 978-1-4503-5620-6.

[34] K. Wongsuphasawat, D. Moritz, A. Anand, M. Jock, B. Howe, and J. Heer, "Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommenda-

tions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 649–658, 2016.

[35] C. Stolte and P. Hanrahan, "Polaris: A system for query, analysis and visualization of multi-dimensional relational databases," in *Proceedings of the IEEE Symposium on Information Vizualization 2000*, ser. INFOVIS '00, Washington, DC, USA: IEEE Computer Society, 2000, pp. 5–14, ISBN: 0-7695-0804-9.

[36] PoleStar. (2016). Polestar, http://vega.github.io/polestar/, (visited on 02/12/2016).

[37] J. Mackinlay, P. Hanrahan, and C. Stolte, "Show me: Automatic presentation for visual analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, 2007.

[38] S. Huron, S. Carpendale, A. Thudt, A. Tang, and M. Mauerer, "Constructive Visualization," in *Proceedings of the conference on Designing interactive systems*, ACM, 2014, pp. 433–442.

[39] F. Bouali, A. Guettala, and G. Venturini, "VizAssist: An Interactive User Assistant for Visual Data Mining," *The Visual Computer*, pp. 1–17, 2015.

[40] D. Gotz and Z. Wen, "Behavior-driven Visualization Recommendation," in *Proceedings of the International Conference on Intelligent User Interfaces*, ser. IUI '09, Sanibel Island, Florida, USA: ACM, 2009, pp. 315–324, ISBN: 978-1-60558-168-2.

[41] D. B. Perry, B. Howe, A. M. Key, and C. Aragon, "Vizdeck: Streamlining Exploratory Visual Analytics of Scientific Data," 2013.

[42] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis, "SEEDB: Automatically Generating Query Visualizations," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1581–1584, 2014.

[43] A. Cypher and D. C. Halbert, *Watch What I Do: Programming by Demonstration*. MIT press, 1993.

[44] D. C. Halbert, "Programming by Example," PhD thesis, University of California, Berkeley, 1984.

[45] H. Lieberman, *Your Wish Is My Command: Programming by Example*. Morgan Kaufmann, 2001.

[46] A. L. Thomaz and C. Breazeal, "Teachable Robots: Understanding Human Teaching Behavior to Build More Effective Robot Learners," *Artificial Intelligence*, vol. 172, no. 6-7, pp. 716–737, 2008.

[47] A. L. Thomaz and C. Breazeal, "Reinforcement Learning with Human Teachers: Evidence of Feedback and Guidance with Implications for Learning Performance," in *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, ser. AAAI'06, Boston, Massachusetts: AAAI Press, 2006, pp. 1000–1005, ISBN: 978-1-57735-281-5.

[48] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A Survey of Robot Learning from Demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[49] J. Lin, J. Wong, J. Nichols, A. Cypher, and T. A. Lau, "End-user Programming of Mashups with Vegemite," in *Proceedings of the International Conference on Intelligent User Interfaces*, ser. IUI '09, Sanibel Island, Florida, USA: ACM, 2009, pp. 97–106, ISBN: 978-1-60558-168-2.

[50] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer, "Wrangler: Interactive Visual Specification of Data Transformation Scripts," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11, Vancouver, BC, Canada: ACM, 2011, pp. 3363–3372, ISBN: 978-1-4503-0228-9.

[51] T. Igarashi and J. F. Hughes, "A Suggestive Interface for 3D Drawing," in *Proceedings of the Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '01, Orlando, Florida: ACM, 2001, pp. 173–181, ISBN: 1-58113-438-X.

[52] M. M. Zloof, "Query by Example," in *AFIPS National Computer Conference*, 1975.

[53] B. A. Myers, in, ch. Peridot: Creating User Interfaces by Demonstration.

[54] B. A. Myers, J. Goldstein, and M. A. Goldberg, "Creating Charts by Demonstration," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '94, Boston, Massachusetts, USA: ACM, 1994, pp. 106–111, ISBN: 0-89791-650-6.

[55] D. Schroeder and D. F. Keefe, "Visualization-by-Sketching: An Artist's Interface for Creating Multivariate Time-Varying Data Visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 877–885, 2016.

[56] J. Walny, B. Lee, P. Johns, N. H. Riche, and S. Carpendale, "Understanding Pen and Touch Interaction for Data Exploration on Interactive Whiteboards," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2779–2788, 2012.

[57] W. O. Chao, T. Munzner, and M. van de Panne, "Poster: Rapid Pen-centric Authoring of Improvisational Visualizations with Napkinvis," *Posters Compendium InfoVis*, vol. 2, no. 1, p. 2, 2010.

[58] A. Endert, P. Fiaux, and C. North, "Semantic Interaction for Visual Text Analytics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12, Austin, Texas, USA: ACM, 2012, pp. 473–482, ISBN: 978-1-4503-1015-4.

[59] J. Liu, E. T. Brown, R. Chang, and C. E. Brodley, "Dis-function: Learning Distance Functions Interactively," in *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, ser. VAST '12, Washington, DC, USA: IEEE Computer Society, 2012, pp. 83–92, ISBN: 978-1-4673-4752-5.

[60] A. Endert, C. Han, D. Maiti, L. House, S. Leman, and C. North, "Observation-level interaction with statistical models for visual analytics," in *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2011, pp. 121–130.

[61] B. Shneiderman, "Direct Manipulation: A Step Beyond Programming Languages," *Computer*, vol. 16, no. 8, pp. 57–69, 1983.

[62] C. Andrews, A. Endert, and C. North, "Space to Think: Large High-resolution Displays for Sensemaking," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2010, pp. 55–64.

[63] S. Huron, Y. Jansen, and S. Carpendale, "Constructing Visual Representations: Investigating the Use of Tangible Tokens," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2102–2111, 2014.

[64] B. Kondo and C. Collins, "DimpVis: Exploring Time-varying Information Visualizations by Direct Manipulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2003–2012, 2014.

[65] Saket B. and Srinivasan A. and Ragan E. D. and Endert A., "Evaluating Interactive Graphical Encodings for Data Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[66] B. Saket, S. Huron, C. Perin, and A. Endert, "Investigating direct manipulation of graphical encodings as a method for user interaction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 482–491, 2020.

[67] H. V. Henderson and P. F. Velleman, "Building Multiple Regression Models Interactively," *Biometrics*, pp. 391–411, 1981.

[68] (2015). Tableau Datasets, https://public.tableau.com/s/resources, (visited on 02/12/2016).

[69] T. Baudel, "From information visualization to direct manipulation: Extending a generic visualization framework for the interactive editing of large datasets," in *Proceedings of the 19th Annual ACM Symposium on User Interface Software and*

Technology, ser. UIST '06, Montreux, Switzerland: ACM, 2006, pp. 67–76, ISBN: 1-59593-313-1.

[70] B. c. Kwon, W. Javed, N. Elmqvist, and J. S. Yi, "Direct manipulation through surrogate objects," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11, Vancouver, BC, Canada: ACM, 2011, pp. 627–636, ISBN: 978-1-4503-0228-9.

[71] M. Bostock, V. Ogievetsky, and J. Heer, "D^3: Data-driven Documents," *IEEE Trans. Visualization & Comp. Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

[72] J. W. Creswell, *Qualitative inquiry and research design: Choosing among five tradition*, 1998.

[73] J. M. Morse, *Determining sample size*, 2000.

[74] J. W. Creswell, *Educational Research: Planning, Conducting, and Evaluating Quantitative*. Prentice Hall Upper Saddle River, NJ, 2002.

[75] S. H. G. G. M. R. P. J. V. S. C. Jagoda Walny Charles Perin, "Constructing personalized tools for collaborative creative work," *To appear*,

[76] M. Beaudouin-Lafon, "Instrumental interaction: An interaction model for designing post-wimp user interfaces," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, ACM, 2000, pp. 446–453.

[77] J. Boy, R. A. Rensink, E. Bertini, and J.-D. Fekete, "A principled way of assessing visualization literacy," *IEEE Transactions on Visualization Computer Graphics*, vol. 20, no. 12, pp. 1963–1972, 2014.

[78] W. S. Cleveland and R. McGill, "Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods," *Journal of the American Statistical Association*, vol. 79, no. 387, pp. 531–554, 1984.

[79] J. Heer and M. Bostock, "Crowdsourcing Graphical Perception: using Mechanical Turk to Assess Visualization Design," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2010, pp. 203–212.

[80] M. Bostock, V. Ogievetsky, and J. Heer, "D^3: Data-driven Documents," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 12, pp. 2301–2309, 2011.

[81] W. S. Cleveland and R. McGill, "Graphical Perception and Graphical Methods for Analyzing Scientific Data," *Science*, vol. 229, no. 4716, pp. 828–833, 1985.

[82] J. Mackinlay, "Automating the Design of Graphical Presentations of Relational Information," *Acm Transactions On Graphics (Tog)*, vol. 5, no. 2, pp. 110–141, 1986.

[83] P. Benner, *Interpretive Phenomenology: Embodiment, Caring, and Ethics in Health and Illness*. Sage publications, 1994.

[84] A. Dix, *Human-Computer Interaction*. Springer, 2009.

[85] E. R. Tufte and P. Graves-Morris, *The Visual Display of Quantitative Information*, 9. Graphics Press Cheshire, CT, 1983, vol. 2.

[86] N. H. Riche, B. Lee, and C. Plaisant, "Understanding interactive legends: A comparative evaluation with standard widgets," in *Computer graphics forum*, Wiley Online Library, vol. 29, 2010, pp. 1193–1202.

[87] M. Beaudouin-Lafon, "Instrumental Interaction: An Interaction Model for Designing post-WIMP User Interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '00, The Hague, The Netherlands: ACM, 2000, pp. 446–453, ISBN: 1-58113-216-6.

[88] L. Grammel, M. Tory, and M.-A. Storey, "How Information Visualization Novices Construct Visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 943–952, 2010.

[89] G. G. Méndez, M. A. Nacenta, and U. Hinrichs, "Considering agency and data granularity in the design of visualization tools," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18, Montreal QC, Canada: ACM, 2018, 638:1–638:14, ISBN: 978-1-4503-5620-6.

[90] M. Tobiasz, P. Isenberg, and S. Carpendale, "Lark: Coordinating co-located collaboration with information visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1065–1072, Nov. 2009.

[91] G. G. Méndez, U. Hinrichs, and M. A. Nacenta, "Bottom-up vs. Top-down: Trade-offs in Efficiency, Understanding, Freedom and Creativity with InfoVis Tools," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI '17, Denver, Colorado, USA: ACM, 2017, pp. 841–852, ISBN: 978-1-4503-4655-9.

[92] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. Mackinlay, B. Howe, and J. Heer, "Voyager 2: Augmenting Visual Analysis with Partial View Specifications," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI '17, Denver, Colorado, USA: ACM, 2017, pp. 2648–2659, ISBN: 978-1-4503-4655-9.

[93] B. Shneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," in *Proceedings of the IEEE Symposium on Visual Languages*, ser. VL '96, Washington, DC, USA: IEEE Computer Society, 1996, pp. 336–, ISBN: 0-8186-7508-X.

[94] L. Ren, J. Cui, Y. Du, and G. Dai, "Multilevel Interaction Model for Hierarchical Tasks in Information Visualization," in *Proceedings of the 6th International Symposium on Visual Information Communication and Interaction*, ser. VINCI '13, Tianjin, China: ACM, 2013, pp. 11–16, ISBN: 978-1-4503-1988-1.

[95] A. Dix and G. Ellis, "Starting Simple: Adding Value to Static Visualisation Through Simple Interaction," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI '98, L'Aquila, Italy: ACM, 1998, pp. 124–134.

[96] B. Saket, A. Srinivasan, E. D. Ragan, and A. Endert, "Evaluating interactive graphical encodings for data visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[97] P. Koytek, C. Perin, J. Vermeulen, E. André, and S. Carpendale, "Mybrush: Brushing and linking with personal agency," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 605–615, 2018.

[98] Z. Pousman, J. Stasko, and M. Mateas, "Casual information visualization: Depictions of data in everyday life," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1145–1152, 2007.

[99] C. Perin, "Direct manipulation for information visualization," PhD thesis, Paris 11, 2014.

[100] J. Vermeulen, K. Luyten, E. van den Hoven, and K. Coninx, "Crossing the Bridge over Norman's Gulf of Execution: Revealing Feedforward's True Identity," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '13, Paris, France: ACM, 2013, pp. 1931–1940, ISBN: 978-1-4503-1899-0.

[101] J. Boy, L. Eveillard, F. Detienne, and J.-D. Fekete, "Suggested Interactivity: Seeking Perceived Affordances for Information Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 639–648, Jan. 2016.

[102] L. Grammel, C. Bennett, M. Tory, and M.-A. Storey, "A survey of visualization construction user interfaces," *EuroVis-Short Papers*, pp. 19–23, 2013.

[103] J. Grudin, "The case against user interface consistency," *Commun. ACM*, vol. 32, no. 10, pp. 1164–1173, Oct. 1989.

[104] K. Börner, A. Maltese, R. N. Balliet, and J. Heimlich, "Investigating aspects of data visualization literacy using 20 information visualizations and 273 science museum visitors," *Information Visualization*, vol. 15, no. 3, pp. 198–213, Jul. 2016.

[105] B. Schwartz, *The paradox of choice: Why more is less*. HarperCollins New York, 2004, vol. 6.

[106] E. D. Ragan, A. Endert, D. A. Bowman, and F. Quek, "The effects of spatial layout and view control on cognitive processing," in *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '11, Vancouver, BC, Canada: ACM, 2011, pp. 2005–2010, ISBN: 978-1-4503-0268-5.

[107] U. Hinrichs and S. Forlini, "In defense of sandcastles: Research thinking through visualization in dh," in *Proceedings of the conference on Digital Humanities*, 2017.

[108] C. Perin, P. Dragicevic, and J. D. Fekete, "Revisiting bertin matrices: New interactions for crafting tabular visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2082–2091, 2014.

[109] B. Nissen and J. Bowers, "Data-things: Digital fabrication situated within participatory data translation activities," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '15, Seoul, Republic of Korea: ACM, 2015, pp. 2467–2476, ISBN: 978-1-4503-3145-6.

[110] H. Lam, "A Framework of Interaction Costs in Information Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1149–1156, 2008.

[111] D. A. Norman, *The design of everyday things: Revised and expanded edition*. Basic books, 2013.

[112] ——, *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*. Basic Books, 1993.

[113] T. Blascheck, L. M. Vermeulen, J. Vermeulen, C. Perin, W. Willett, T. Ertl, and S. Carpendale, "Exploration strategies for discovery of interactivity in visualizations," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018.

[114] A. Srinivasan, M. Dontcheva, E. Adar, and S. Walker, "Discovering natural language commands in multimodal interfaces," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, ser. IUI '19, Marina del Ray, California: ACM, 2019, pp. 661–672, ISBN: 978-1-4503-6272-6.

[115] M. Turk, "Multimodal interaction: A review," *Pattern Recognition Letters*, vol. 36, pp. 189–195, 2014.

[116] R. Nugent and M. Meila, "An overview of clustering applied to molecular biology," in *Statistical methods in molecular biology*, Springer, 2010, pp. 369–404.

[117] G. M. Downs and J. M. Barnard, "Clustering methods and their uses in computational chemistry," *Reviews in computational chemistry*, vol. 18, pp. 1–40, 2002.

[118] D. J. Bartholomew, F. Steele, J. Galbraith, and I. Moustaki, *Analysis of multivariate social science data*. Chapman and Hall/CRC, 2008.

[119] M. desJardins, J. MacGlashan, and J. Ferraioli, "Interactive visual clustering," in *Proceedings of the 12th International Conference on Intelligent User Interfaces*, ser. IUI '07, Honolulu, Hawaii, USA: ACM, 2007, pp. 361–364, ISBN: 1-59593-481-2.

[120] M. Cavallo and Demiralp, "Clustrophile 2: Guided visual clustering analysis," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018.

[121] K. Chen and L. Liu, "Vista: Validating and refining clusters via visualization," *Information Visualization*, vol. 3, no. 4, pp. 257–270, Dec. 2004.

[122] J. Seo and B. Shneiderman, "Interactively exploring hierarchical clustering results [gene identification]," *Computer*, vol. 35, no. 7, pp. 80–86, 2002.

[123] N. Cao, D. Gotz, J. Sun, and H. Qu, "Dicon: Interactive visual analysis of multidimensional clusters," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2581–2590, 2011.

[124] D. H. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang, "Ipca: An interactive system for pca-based visual analytics," in *Proceedings of the 11th Eurographics / IEEE - VGTC Conference on Visualization*, ser. EuroVis'09, Berlin, Germany: The Eurographs Association & John Wiley & Sons, Ltd., 2009, pp. 767–774.

[125] B. C. Kwon, B. Eysenbach, J. Verma, K. Ng, C. D. Filippi, W. F. Stewart, and A. Perer, "Clustervision: Visual supervision of unsupervised clustering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 142–151, 2018.

[126] A. Endert, L. Bradel, and C. North, "Beyond control panels: Direct manipulation for visual analytics," *IEEE Computer Graphics and Applications*, vol. 33, no. 4, pp. 6–13, 2013.

[127] M. Sedlmair, M. Meyer, and T. Munzner, "Design study methodology: Reflections from the trenches and the stacks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2431–2440, 2012.

[128] (2018). GWAS Catalog, https://www.ebi.ac.uk/gwas/, (visited on 02/12/2018).

[129] (2018). R Language, https://www.r-project.org/about.html, (visited on 02/12/2016).

[130] (2018). SAS Software, https://www.sas.com, (visited on 02/12/2016).

[131] S. L'Yi, B. Ko, D. Shin, Y.-J. Cho, J. Lee, B. Kim, and J. Seo, "Xclusim: A visual analytics tool for interactively comparing multiple clustering results of bioinformatics data," *BMC Bioinformatics*, vol. 16, no. 11, S5, 2015.

[132] kern, A. Lex, N. Gehlenborg, and C. R. Johnson, "Interactive visual exploration and refinement of cluster assignments," *BMC Bioinformatics*, vol. 18, no. 1, p. 406, 2017.

[133] E. Horvitz, "Principles of mixed-initiative user interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '99, Pittsburgh, Pennsylvania, USA: ACM, 1999, pp. 159–166, ISBN: 0-201-48559-1.

[134] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[135] Python Core Team (2018), *Python: A dynamic, open source programming language*, Python Software Foundation, Vienna, Austria, 2018.

[136] S. Zolaktaf and G. Murphy, "What to learn next: Recommending commands in a feature-rich environment," Dec. 2015, pp. 1038–1044.

[137] E. Reed, S. Nunez, D. Kulp, J. Qian, M. P. Reilly, and A. S. Foulkes, "A guide to genome-wide association analysis and post-analytic interrogation," *Statistics in Medicine*, vol. 34, no. 28, pp. 3769–3792, 2015. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/sim.6605`.

[138] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[139] *Sk learn - clustering metrics*, `https://scikit-learn.org/stable/modules/classes.html`, Accessed: 2019-07-15.

[140] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

[141] A. Malhi and R. X. Gao, "Pca-based feature selection scheme for machine defect classification," *IEEE Transactions on Instrumentation and Measurement*, vol. 53, no. 6, pp. 1517–1525, 2004.

[142] Ç. Demiralp, "Clustrophile: A tool for visual clustering analysis," *CoRR*, vol. abs/1710.02173, 2017. arXiv: 1710.02173.

[143] M. Gleicher, "Considerations for visualizing comparison," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 413–423, 2018.

[144] B. Saket and A. Endert, "Demonstrational interaction for data visualization," *IEEE Computer Graphics and Applications*, vol. 39, no. 3, pp. 67–72, 2019.

[145] B. Saket, H. Kim, E. T. Brown, and A. Endert, "Visualization by Demonstration: An Interaction Paradigm for Visual Data Exploration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 331–340, 2017.

[146] B. Saket and A. Endert, "Investigating the manual view specification and visualization by demonstration paradigms for visualization construction," *Computer Graphics Forum*, 2019.

[147] B. Saket, L. Jiang, C. Perin, and A. Endert, *Liger: Combining interaction paradigms for visual analysis*, arXiv preprint 1907.08345, 2019.

[148] D. A. Norman, *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1993.